INPUT

Publicación práctica para usuarios de

sinclair

Revista m

Precio 350 Ptas

Año 2 Número 13

DESENREDA TUS CADENAS

MAS ENTORNO A LAS INTERRUPCIONES

AGREGALE INSTRUCCIONES AL BASIC

PUZZLES Y MATEMATICAS

LISTA DE PREMIADOS EN EL SORTEO DE VERANO

¿LO HUBIERA PODIDO COMPRAR MAS BARATO?

Los clientes de Regisa esta pregunta ya no se la hacen. Pero además cuando conozcan las **nuevas ofertas** de monitores, ordenadores, impresoras, unidades de disco, periféricos, software, etc. **(evidentemente todo con garantía),** que ha preparado Regisa, se van a llevar una agradable sorpresa.

ventas al mayor

REGISA

Comercio, 11 - Tel. 319 93 08 - Barcelona

lo mismo y más..., pero al mejor precio.

sinclair AMSTRAD SPECTRAVIDEO SEIKOSHA DK-TRONIC

commodore HIT BIT SONY RITEMAN FONTEC

Establecimientos recomendados: • BAZAR DELHI. Reina Cristina, 11. Barcelona • INTERJOYA. Reina Cristina, 9. Barcelona • BAZAR TAIWAN. Plaza Palacio, 19 (Galerías). Barcelona • LOS GUERRILLEROS. I. Canarias, 128. Valencia • BAZAR KARDIS. I. Canarias, 130. Valencia • BAZAR DELHI. M. Ruano, 5. Lleida

MONTERO & ASOCIADOS

AÑO 2 NUMERO 13

DIRECTOR:
Alejandro Diges
COORDINADOR EDITORIAL:
Francisco de Molina
DISEÑO GRAFICO:
Tomás López
COLABORADORES:
Antonio Taratiel, Luis R. Palencia, Francisco Tórtola, Benito Román, Esther de la Cal, Ernesto del Valle, Equipo Molisoft.

INPUT Sinclair es una publicación juvenil de EDICIONES FORUM

GERENTE DIVISION DE REVISTAS:
Angel Sabat

PUBLICIDAD: José Real-Grupo Jota
Madrid: c/ General Varela, 35
Teléf. 270 47 02/03
Barcelona: Avda. de Sarriá, 11-13, 1.º
Teléf. 250 23 99

FOTOMECANICA: Ochoa, S. A.

COMPOSICION: EFCA, S. A.

IMPRESION: Sirven Grafic
C/ Gran Vía, 754-756. 08013 Barcelona
Depósito legal: B-21954-1986

SUSCRIPCIONES: EDISA,
López de Hoyos, 141. 28002 Madrid
Teléf. (91) 415 97 12

REDACCION:
Paseo de la Castellana, 93, 14.ª
28046 Madrid. Teléf. 456 54 13

DISTRIBUIDORA
R.B.A. PROMOTORA DE EDICIONES, S. A.
Travesera de Gracia, 56. Edificio Odiseus.
08006 Barcelona.

El precio será el mismo para Canarias que para la Península y en él irá incluida la sobretasa aérea.

INPUT Sinclair es una publicación controlada por

INPUT Sinclair es independiente y no está vinculada a Sinclair Research o sus distribuidores.

INPUT no mantiene correspondencia con sus lectores, si bien la recibe, no responsabilizándose de su pérdida o extravío. Las respuestas se canalizarán a través de las secciones adecuadas en estas páginas.

INPUT sinclair

SUMARIO

NUEVAS TENDENCIAS

Abrumadora ha sido realmente la respuesta al sorteo que propusimos en el número especial de verano. Ahora llega el momento de comprobar si vuestro nombre aparece en la relación de afortunados. Si estáis incluidos, entonces enhorabuena. De lo contrario no desesperéis, es intención de quienes hacemos **INPUT** continuar ofreciendo la posibilidad de obtener algún premio lo más a menudo que permitan las circunstancias. Este es el caso del ganador en el concurso convocado por **Anaya Multimedia** desde la revista. Como resultado acudió a la feria **PCW** en Londres uno de los lectores más habilidosos descifrando claves. Según cuentan las crónicas se divirtió de lo lindo. Septiembre una vez más ha sido pródigo en la aparición de rumores y nuevos productos. La feria británica antes mencionada junto con el **Sonimag** han servido para apreciar cómo parecen confirmarse dos tendencias de cara al futuro de la microinformática. Por un lado los ordenadores destinados a juegos con más memoria, capacidades gráficas y sonido, siendo buena muestra el nuevo **Spectrum+2** y los **MSX 2.** Por otro lado, los ordenadores compatibles **PC** de bajo precio de venta, situándose al alcance de muchos más usuarios, sistemas que hasta hace poco eran un lujo asiático para quienes no fueran empresas o profesionales. El denominador común de ambas tendencias es la bajada de precios aplicada en septiembre. De lo que no cabe duda es que nos encontramos ante el umbral de una nueva etapa.

LOS MEJORES DE INPUT

Hemos pensado que es interesante disponer de un ***ranking*** que ponga en claro, mes a mes, cuáles son los programas preferidos de nuestros lectores. Para ello, es obligado preguntaros directamente y tener así el mejor termómetro para conocer vuestras preferencias. Podéis votar por cualquier programa aunque no haya sido comentado todavía en **INPUT**.

El resultado de las votaciones será publicado en cada número de **INPUT**.

Entre los votantes sortearemos 10 cintas de los títulos que pidáis en vuestros cupones.

Nota: No es preciso que cortéis la revista, una copia hecha a máquina o una simple fotocopia sirven.

Enviad vuestros votos a: **LOS MEJORES DE INPUT** P.º de la Castellana, 93. Planta 14. 28046 Madrid

ELIGE TUS PROGRAMAS

Primer título elegido _____ **Segundo título elegido** _____

Tercer título elegido _____ **Programa que te gustaría conseguir** _____

Qué ordenador tienes _____ **Nombre** _____

1.er Apellido _____ **2.º Apellido** _____

Fecha de nacimiento _____ **Teléfono** _____

Dirección _____ **Localidad** _____

Provincia _____

MUSICA Y TECNOLOGIA

Acaba de llegar a nuestras manos el cuarto número de la revista especializada "Música y Tecnología". Su contenido es denso y cuidado, tratando gran variedad de témas que han de interesar a quienes consideran que saber de música no es solo dominar un instrumento.

****

RELEVO EN GALERIAS

Thomas Enders, quien desde su cargo como jefe de compras de la división "ON-line" dió el empujón definitivo a la microinformática en los almacenes Galerías Preciados, acaba de regresar a los EE.UU. para ampliar sus estudios en la universidad de Yale.
El distribuidor de software SERMA no lo dudó un momento, haciendole entrega de una placa de agradecimiento por su obra, acto al que acudió una representativa muestra de la prensa especializada.

****

NUEVA LINEA INVES

Investrónica, la empresa que implantó los ordenadores de Sinclair en nuestro país, acaba de optar por una nueva estrategia en su intento de poner la electrónica de consumo al alcance de todos.
Por un lado entra en los equipos de sonido con una cadena llamada 100 HF, que puede ver sus prestaciones ampliadas en base a un disco compacto y el sintonizador digital en el modelo CD 300 HF, aunque también existe el compact disk por separado, DISC CD 200.
Por otro lado entra en el mundo de los ordenadores compatibles con el estándar PC con tres modelos: INVES 256X, PC 640A y PC 640X.
El denominador común de toda la línea INVES es su bajo precio de venta en realación con sus prestaciones.
La entrada de estas grandes firmas distribuidoras parece que vienen a hacer realidad el gran sueño de sir Clive Sinclair, quien hace tiempo lanzó equipos de HIFI a bajo precio en el Reino Unido, aunque utilizando la tecnología mas limitada de aquel momento.

****

EL PLUS DOS VIO LA LUZ

El esperado ZX Spectrum Plus 2 por fin ha sido presentado en sociedad. En primer lugar se mostró a principios de septiembre en el PCW de Londres y unos días más tarde llegó al Sonimag de Barcelona.
Las primeras novedades que se aprecian son un teclado similar al utilizado por los modelos de Amstrad, así como la unidad de cassette incorporada en la misma carcasa y los ports para joysticks. Dispone de tres canales se sonido (como el 128) y una voz generada por el microprocesador. Incluye también el interface musical MIDI.
Existe un port serie a través del cual se puede conectar una impresora estándar. Igualmente puede atacar un monitor de color mediante la salida RGB.
La memoria RAM es de 128 Kbytes y 32 Kbytes para la ROM. El microprocesador sigue siendo el clásico Z80A.
Su compatibilidad con los programas desarrollados para el modelo de 48 K está asegurada.
Trabaja con una versión ampliada del BASIC.
Seguiremos informando mas detalladamente cuando el ordenador llegue a nuestras manos.
Sinclair adopta en este modelo el llamado SQC (Sinclair Quality Control), que confirma que el programa ha sido revisado por los técnicos de Sinclair para asegurar su total garantía de funcionamiento.

****

RELACIONES LOGICAS

Teclea para Spectrum ZX81

- OPERADORES RELACIONALES
- ¿UNA COSA ES MAYOR, IGUAL O MENOR QUE OTRA?
- OPERADORES LOGICOS
- USO DE AND, OR Y NOT

Los ordenadores son capaces de ejecutar millones de operaciones lógicas por segundo. Pero la lógica es además una parte fundamental de cualquier programa.

Hay tres tipos de expresiones que se utilizan en la programación con BASIC. Una gran parte de todo programa está constituída por operaciones aritméticas y las realizadas con cadenas de caracteres; sin embargo, es el tercer tipo de operaciones, las realizadas con las expresiones lógicas, el que realmente se ocupa de la toma de decisiones dentro de un programa.

Su función elemental consiste en evaluar si algo es «verdadero» o «falso». Las palabras y símbolos que utiliza el programa para hacer esto se llaman operadores (en algunos casos se llaman también conectores.

En las expresiones lógicas se utilizan dos tipos de operadores: los relacionales (que incluyen símbolos matemáticos tales como <, > e =), y los lógicos que forman parte del repertorio de palabras reservadas en todas las versiones del BASIC: AND, OR y NOT.

MAYOR, IGUAL O MENOR

Los operadores relacionales pueden utilizarse incluso en los programas BASIC más sencillos. Las posibles comparaciones que utilizan estos operadores son:

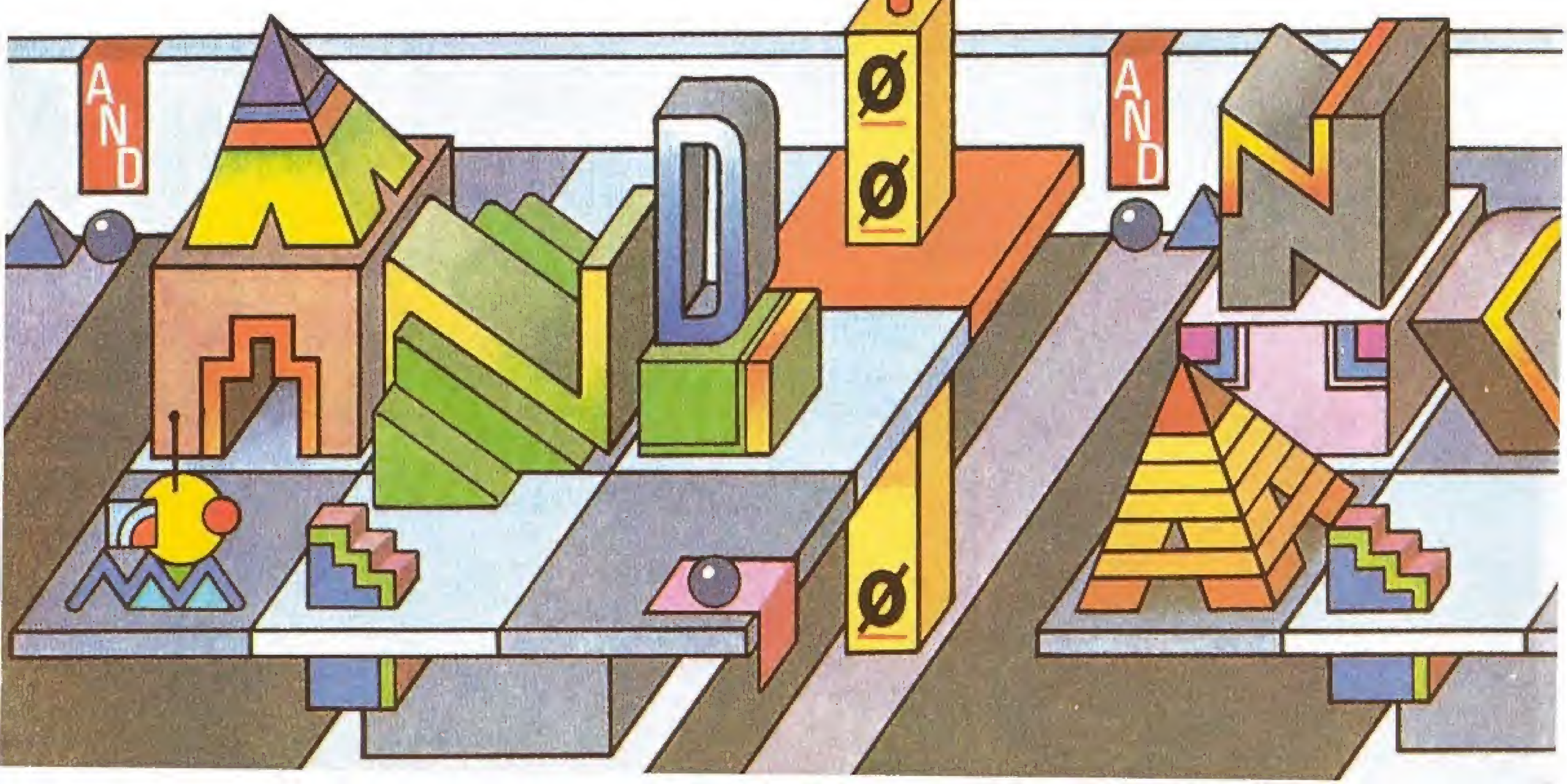
A > B ... A es mayor que B
A < B ... A es menor que B
A ⩾ B ... A es mayor o igual que B
A ⩽ B ... A es menor o igual que B
A = B ... A es igual que B
A <> B ... A es diferente de B

Habrás visto estos operadores utilizados docenas de veces en los programas de INPUT o en cualquier otra parte, y aunque su utilización es múltiple, su valor en las pruebas condicionales resulta más evidente en relación con la sentencia IF ... THEN. Un ejemplo típico de esto es:

```
IF A>B THEN PRINT "A ES
MAYOR QUE B"
```

En este caso, el valor de la variable A debe superar el valor de B para que se ejecute el resto de la línea. En los casos en que el valor de A permanezca por debajo del valor de B, el programa se limitará a pasar a la línea siguiente. Puedes ver que es posible disponer de una cierta capacidad de salto condicional: si resulta un determinado conjunto de valores, el programa hace una cosa; si resulta otro conjunto de valores, el programa hace otra cosa distinta.

Los operadores relacionales no están únicamente limitados a manejar valores numéricos; también sirven para comparar cadenas de caracteres. No obstante, hay que manejarlos con precaución para evitar encontrarte con resultados inesperados. Se debe tener presente que la comparación siempre se detiene en el número de caracteres correspondiente a la más corta de las dos cadenas que se compa-



ran. Así, si una de las cadenas contiene once caracteres, mientras que la otra sólo siete, únicamente intervendrán en la comparación los primeros siete caracteres de la cadena más larga, comparándose con sus correspondientes en la cadena corta.

Los caracteres se comparan uno por uno y de izquierda a derecha; de aquí pueden surgir los resultados inesperados. En una comparación de tipo numérico, la proposición 5 > 10 es evidentemente falsa, pero en una comparación entre cadenas, puedes encontrarte una sentencia en la que se comparen dos variables en la siguiente forma: A$ > B$. Si A$ = «5» y B$ = «10» el ordenador compara en primer lugar el carácter que figura más a la izquierda, que en este caso es un 5 en una de las cadenas y un 1 en la otra, y a continuación se para porque considera que ya no tiene nada que comparar.

En consecuencia se obtiene un «verdadero» como resultado de la comparación, debido a que 5 es mayor que 1, pero el ordenador ignora los números restantes de la cadena más larga. Tienes que estar atento para no cometer errores de este tipo.

En las comparaciones entre cadenas de caracteres, a las letras del alfabeto se les supone la siguiente relación de orden: «A» < «B» < «C» < «D» , etcétera. Pero también aquí has de tener cuidado, ya que el ordenador no entiende realmente lo que estos caracteres significan. Lo que hace es comparar los códigos correspondientes a cada carácter, siendo igualmente significativos dentro de la cadena los espacios y otros caracteres que no son letras, pero que disponen también de un código propio. Como puedes imaginarte, esto podría causar estragos en un programa de clasificación de cadenas por orden alfabético.

Cuando ambas cadenas contienen una secuencia de caracteres parcialmente idéntica, se considera numéricamente mayor a la cadena más larga. Pero observa que una cadena más corta es considerada mayor en una expresión como la siguiente: «ABD» > «ABCD» ya que la comparación se detiene en cuanto se encuentra con la primera diferencia, es decir, al comparar los códigos correspondientes a los caracteres «D» y «C». El funcionamiento es de hecho el mismo que se sigue al asignar prioridades alfabéticas en la redacción de un índice o un diccionario, en el que «para» figura antes que «paradoja», pero después de «pan».

VERDADERO O FALSO

Toda comparación produce un resultado que en realidad es un número entero. Cuando la comparación resulta ser falsa, el resultado es 0, mientras que cuando es verdadera el resultado es 1, dependiendo del ordenador que sea. Teclea este comando en modo directo para observar el resultado que corresponde a tu ordenador:

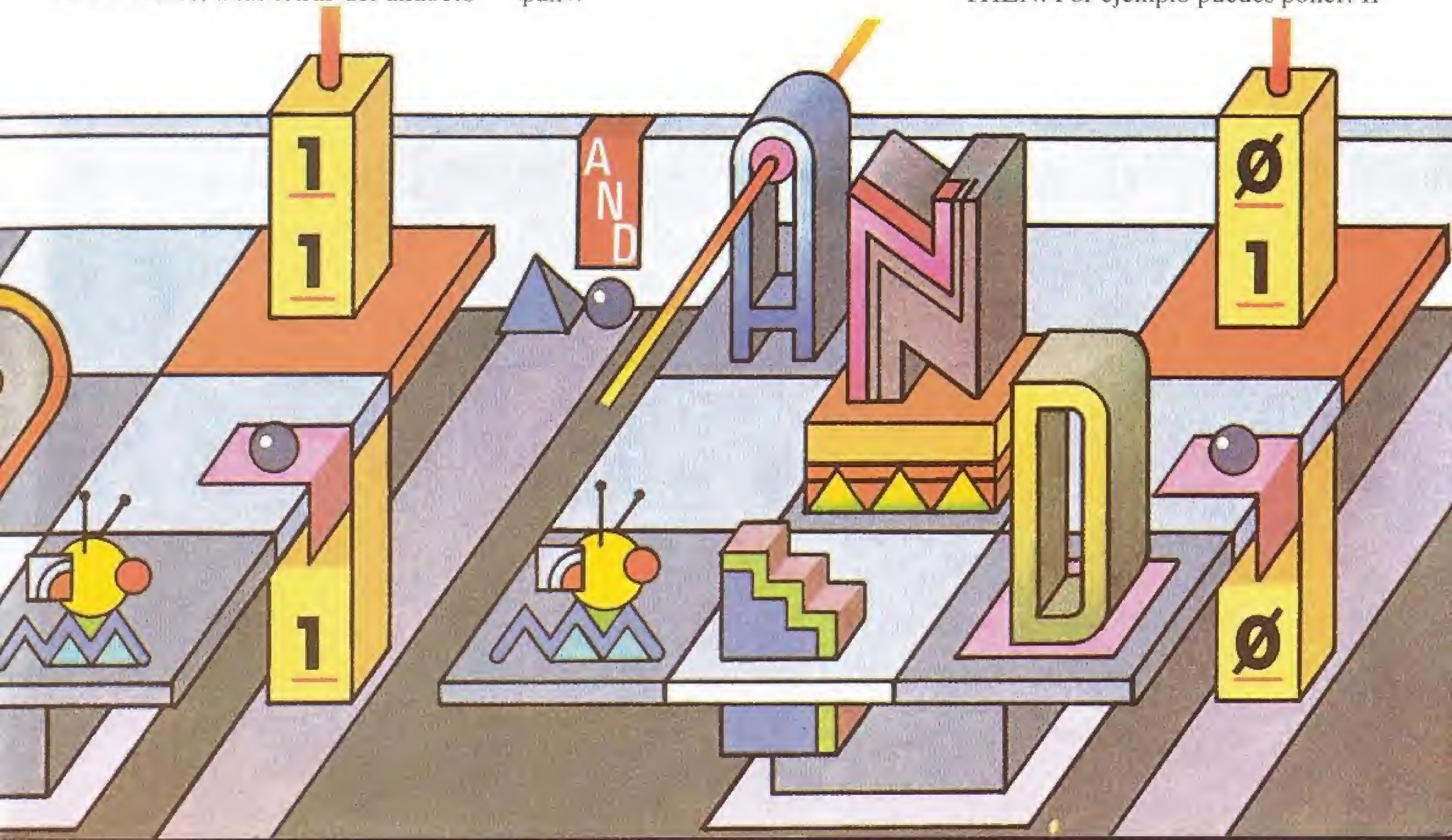
```
PRINT 6>5, 5>7
```

La expresión de la izquierda es verdadera y la de la derecha falsa. En consecuencia obtendrás en la pantalla un 1 o un 0.

Los números enteros que obtengas como resultado de las comparaciones, puedes utilizarlos para realizar cálculos dentro de tus programas. No obstante tienes que tener cuidado de no intentar hacer divisiones por 0, lo que te daría un mensaje de error.

OPERADORES LOGICOS

Las **palabras reservadas** AND, OR y NOT son operadores lógicos que constituyen una poderosa extensión a la capacidad de toma de decisiones por medio de la sentencia IF ... THEN. Por ejemplo puedes poner: IF



esto es cierto AND eso otro es cierto. THEN hacer una determinada cosa. De esta forma puedes ir construyendo funciones de gran complejidad. Estos operadores, que a veces se llaman también operadores **booleanos** (aunque no debes dejarte impresionar por este nombre) se pueden utilizar en comandos directos o dentro de los programas para obtener un «valor de verdad» en condiciones de prueba que pueden llegar a ser muy complejas. De esta forma constituyen una alternativa muy cómoda a lo que en otro caso sería un confuso conglomerado de saltos condicionales a base de una utilización masiva de sentencias IF ... THEN.

USO DE «AND»

En su forma más simple, se puede considerar que el operador AND tiene el mismo significado que la conjunción española «y». En una expresión tal como la siguiente:

```
IF V>0 AND V<100 PRINT
"DENTRO DE RANGO"
```

El mensaje sólo aparecerá en pantalla cuando la variable V esté comprendida entre 0 y 100.

En un programa puedes tener una línea como ésta:

```
990 OKAY=1 AND MES>5 AND
MES<10 AND ANO>1900 AND
ANO<2001
```

Esta línea de programa te permitiría comprobar si una determinada fecha corresponde a un verano del siglo XX.

En este caso se utiliza AND para realizar cuatro pruebas, cuyos resultados deben ser todos ciertos para que OK siga siendo verdadera. En esta prueba de validación se asigna en primer lugar a la variable OK el valor «verdadero». A continuación se impone la condición de que MES caiga en el intervalo de 6 a 9, y de que AÑO esté entre 1901 y 2000. Pero miremos un poco más en detalle lo que ocurre. Si se cumplen todas las condiciones, todas las expresiones darán un valor verdadero. Así, de hecho la línea de programa se convierte en :

```
990 OKAY=-1 AND -1 AND -1
AND -1 AND -1
```

Si ahora le añades PRINT OK, puedes comprobar que el resultado es en efecto 1, es decir verdadero.

USO DE «OR»

El operador OR puede recibir una consideración en gran parte análoga a la de AND, aunque su significado es diferente, como ocurre con la conjunción española «o» en su uso cotidiano.También aquí observaremos su empleo en una sencilla línea de programa en la que se utiliza para comparar los valores de una variable:

```
IF V=8 OR V=10 PRINT "OKAY"
```

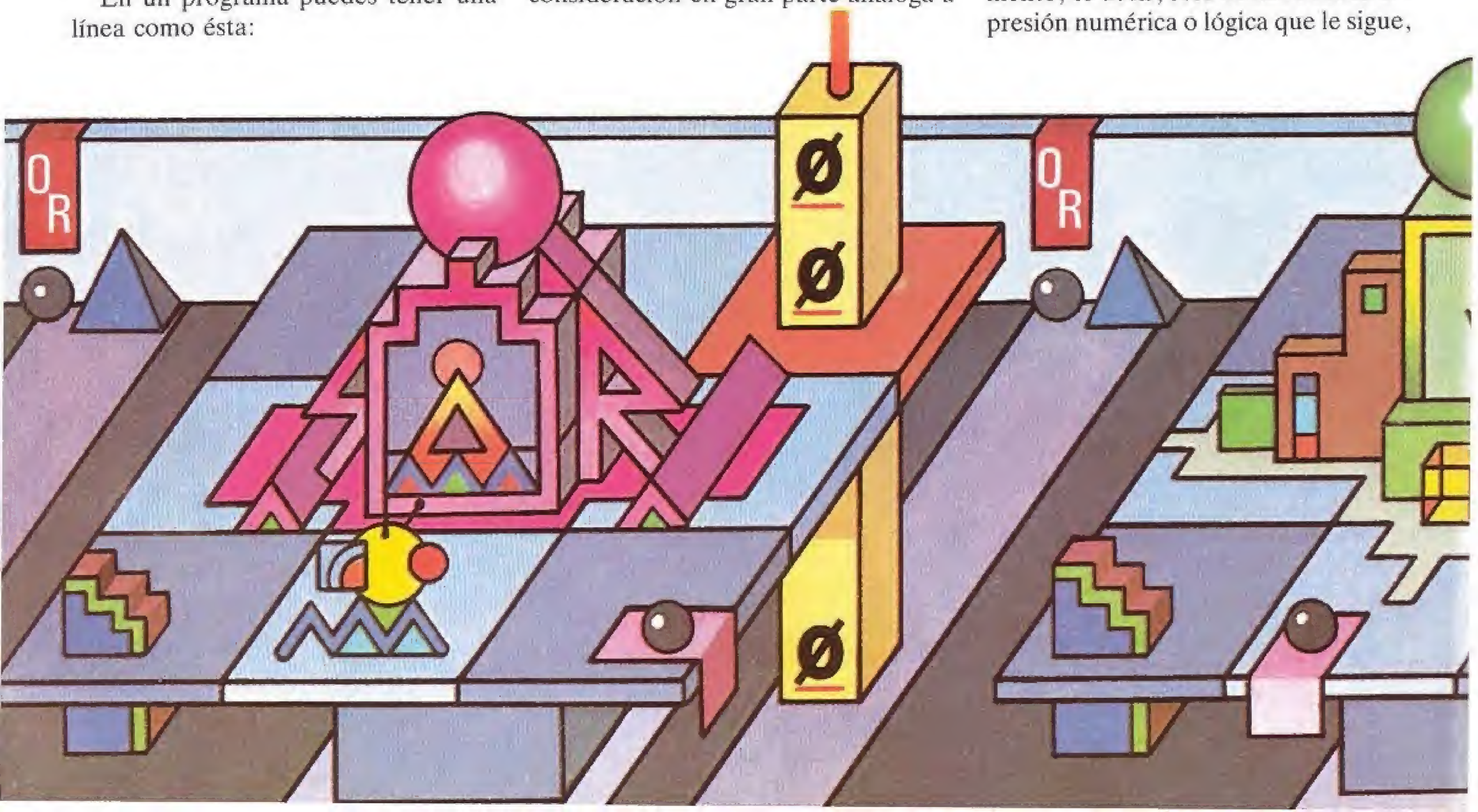
El mensaje aparecerá en la pantalla si el valor de la variable V es 8 o 10.

El operador OR resulta de gran utilidad cuando se comprueba la validez de las entradas suministradas a una sentencia INPUT dentro de un programa. Si por ejemplo el programa te pide que introduzcas tu edad, siempre habrá alguien que tecleará —10 ó 999 con la intención de probar a ver qué pasa. Pero puedes solucionar el problema de la siguiente forma:

```
10 INPUT A
20 IF A<1 OR A>120 THEN
PRINT "ESTAS DE BROMA?":
GOTO 10
```

USO DE «NOT»

El tercer operador lógico que se utiliza con frecuencia es el NOT. Es un poco diferente de los anteriores en el sentido de que sólo utiliza un argumento, es decir, sólo actúa sobre la expresión numérica o lógica que le sigue,



a diferencia de AND y OR que actúan sobre la expresión que les precede y sobre la que les sigue, es decir son operadores binarios o de dos argumentos.

Puedes utilizar NOT en un programa como el siguiente:

```
IF NOT (A>10) THEN 999
```

En el lenguaje hablado normal, podríamos traducir la anterior expresión más o menos así: «Si el valor de A no es mayor que 10, pasa a la línea 999».

La función lógica que realiza el operador NOT es convertir una condición verdadera en falsa y viceversa. Puedes utilizarla pues para invertir el valor resultante de una prueba anterior.

TABLAS DE VERDAD

En relación con los operadores lógicos, te encontrarás con mucha frecuencia con las «tablas de verdad».

En realidad son algo muy sencillo que se utiliza para dar una representación visual de lo que ocurre cuando se hacen operaciones de AND y OR con los valores 1 y 0 (verdadero y falso). El operador NOT no necesita en realidad una tabla de verdad, ya que basta con invertir los valores para observar su efecto.

Las tablas de verdad pueden adoptar formas diferentes. Una forma típica para el operador AND es:

```
 A    B    C
-------------------------
-1   -1   -1     (linea 1)
-1    0    0     (linea 2)
 0   -1    0     (linea 3)
 0    0    0     (linea 4)
```

El significado de la línea 1 de la tabla es el siguiente: «Si A es verdadera y B es verdadera, entonces C es verdadera». El significado de la línea 2 es: «Si A es verdadera y B es falsa, entonces C es falsa». La línea 3 significa: «Si A es falsa y B es verdadera, entonces C es falsa». La línea 4 por su parte significa: «Si A y B son falsas, entonces C es falsa». Análogamente la tabla de verdad del operador OR es:

```
 A    B    C
-------------------------
-1   -1   -1
-1    0   -1
 0   -1   -1
 0    0    0
```

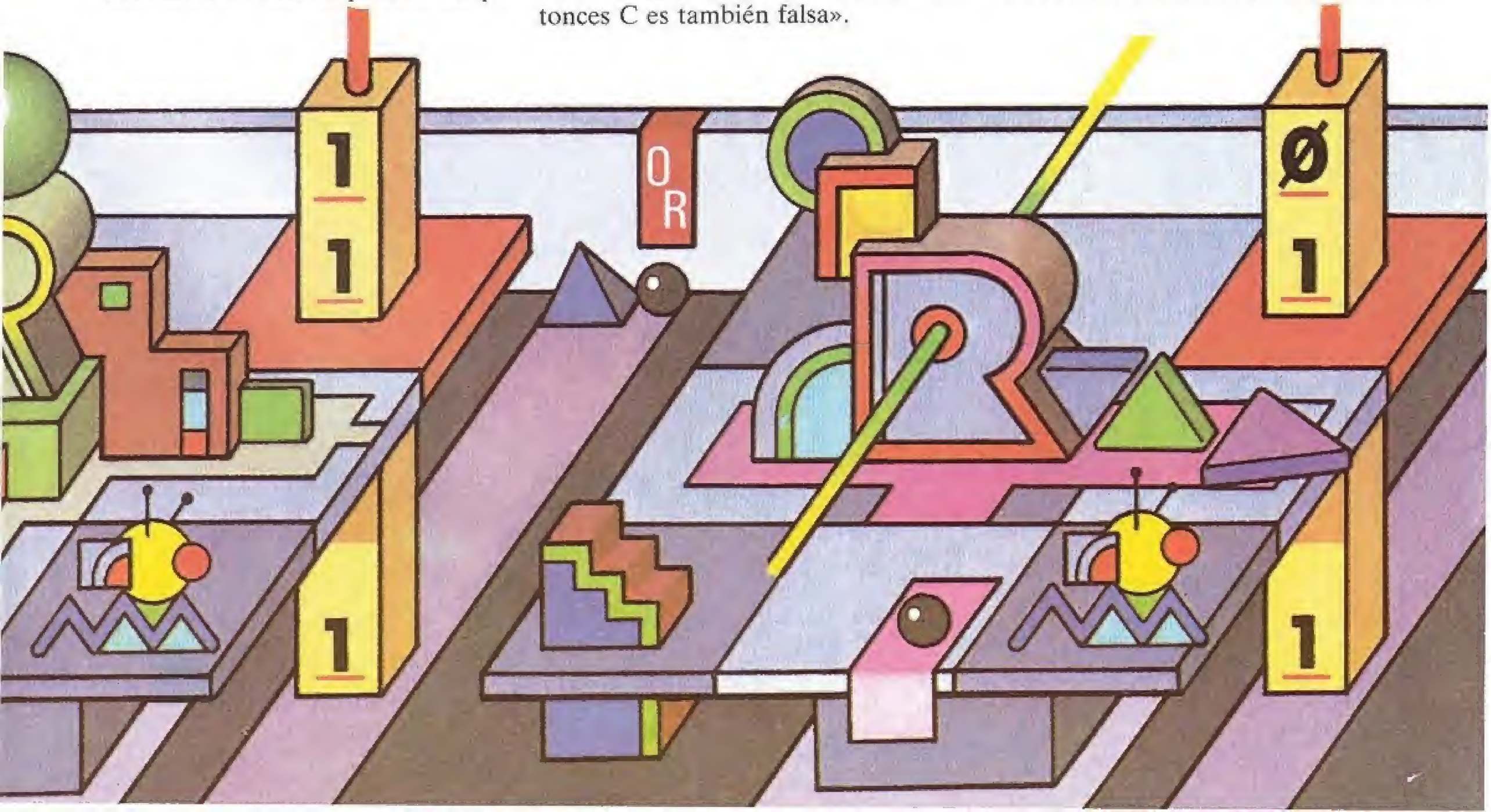
La primera línea significa: «Si A es verdadera y B es verdadera, entonces C es verdadera». Las líneas 2 y 3 pueden ser ambas interpretadas como: «Si A o B son verdaderas, entonces C es verdadera». El significado de la línea 4 es: «Si A y B son ambas falsas, entonces C es también falsa».

APLICACION PRACTICA

Aquí tienes finalmente un programa que utiliza los operadores AND, OR y NOT. Se trata de una versión simplificada de un programa que calcula la escala correcta de salarios para alguien que ha solicitado un empleo:

```
10 INPUT "EDAD";EDAD
20 INPUT "NUMERO DE NIVELES"
   ;NIVELES
30 LET MAS18=(EDAD>=18)
40 LET CUAL=(NIVELES>=5)
50 IF NOT MAS 18 AND NOT
   CUAL THEN PRINT "NO
   UTILIZABLE"
60 IF (NOT MAS18 AND CUAL)
   OR (MAS18 AND NOT CUAL)
   THEN PRINT "ESCALA
   SALARIAL UNO"
70 IF MAS18 AND CUAL THEN
   PRINT "ESCALA SALARIAL
   DOS"
```

Supongamos que como respuesta a las dos primeras sentencias INPUT, tecleas 20 años de edad y cuatro niveles O. Esto significa que (EDAD ⩾ 18) es verdad, mientras que (ONIVEL ⩾ 5) es falso. En consecuencia la persona es MAS18 (más de 18 años) y NO CUAL (no cualificada). El ordenador



encuentra enseguida este conjunto de condiciones en la línea 60, imprimiendo a continuación «ESCALA DE SALARIO UNO». Puedes ampliar fácilmente este programa para que compruebe más condiciones, por ejemplo si el solicitante cuenta con más de dos años de experiencia, o cualquier otra circunstancia que sea necesaria para el trabajo.

OPERACIONES NUMERICAS

El uso de los operadores lógicos no está únicamente limitado a las sentencias IF ... THEN que acabamos de ver. También se pueden utilizar en algunos tipos de operaciones numéricas. Puede que ya las hayas visto utilizadas así en algún programa y a lo mejor te has quedado maravillado de lo que sucedía. De hecho no siempre funcionan de la forma obvia que podría esperarse. Teclea lo siguiente:

```
PRINT 375 AND 47
```

Tal vez esperas que el resultado sea 422, o quizás 37547. Y sin embargo el resultado es 39. ¿Qué ha sucedido? Para explicarlo, tenemos que adentrarnos algo más en la forma de trabajar del ordenador, a fin de entender el modo en que éste maneja los dos argumentos cuyo AND se calcula.

En principio ambas expresiones (375 y 47) se convierten en su equivalente binario de dos bytes.

375 en binario es:
0000000101110111
Por su parte, 47 en binario es:
0000000000101111

A continuación la función AND realiza una comparación bit a bit sobre los 16 bits, dando un 1 como resultado únicamente cuando hay un 1 en el mismo bit de ambos números en binario. Empezando por la derecha, sólo los pares de bits cero, 1, 2 y 5 cumplen esta condición. Esto da como resultado el número binario 0000000000100111.

Al volver a hacer la conversión a decimal, los bits 5, 2, 1 y 0 puestos a 1 dan un valor combinado de 39; éste es el resultado de hacer un AND de 375 y 47.

Ensaya ahora el siguiente comando:

```
PRINT 375 OR 47
```

¿A qué se debe el resultado 383 que se obtiene en este caso? Al realizar la operación con OR lógico el resultado de cada comparación bit a bit es 1 si hay un 1 en cualquiera de los bits del par que se compara.Observa de nuevo la forma binaria de los números anteriores y verás que los bits 8, 6, 5, 4, 3, 2, 1 y 0 tienen un 1 en al menos uno de los bits del par. El número binario resultante es 0000000101111111 que es 383 en decimal.

Con el operador OR es posible obtener el mismo valor de bits con expresiones diferentes. Por ejemplo, si tecleas PRINT 315 OR 75, el resultado es también 383. Si quieres puedes comprobarlo realizando la conversión de los números a forma binaria.

El valor −1 (que se almacena como FFFFFFFF en hexadecimal, es decir un conjunto de bits que son todos 1) queda inalterado cuando con él se realiza una operación de OR lógico con otro número cualquiera. Para hacer la prueba, teclea el siguiente comando directo:

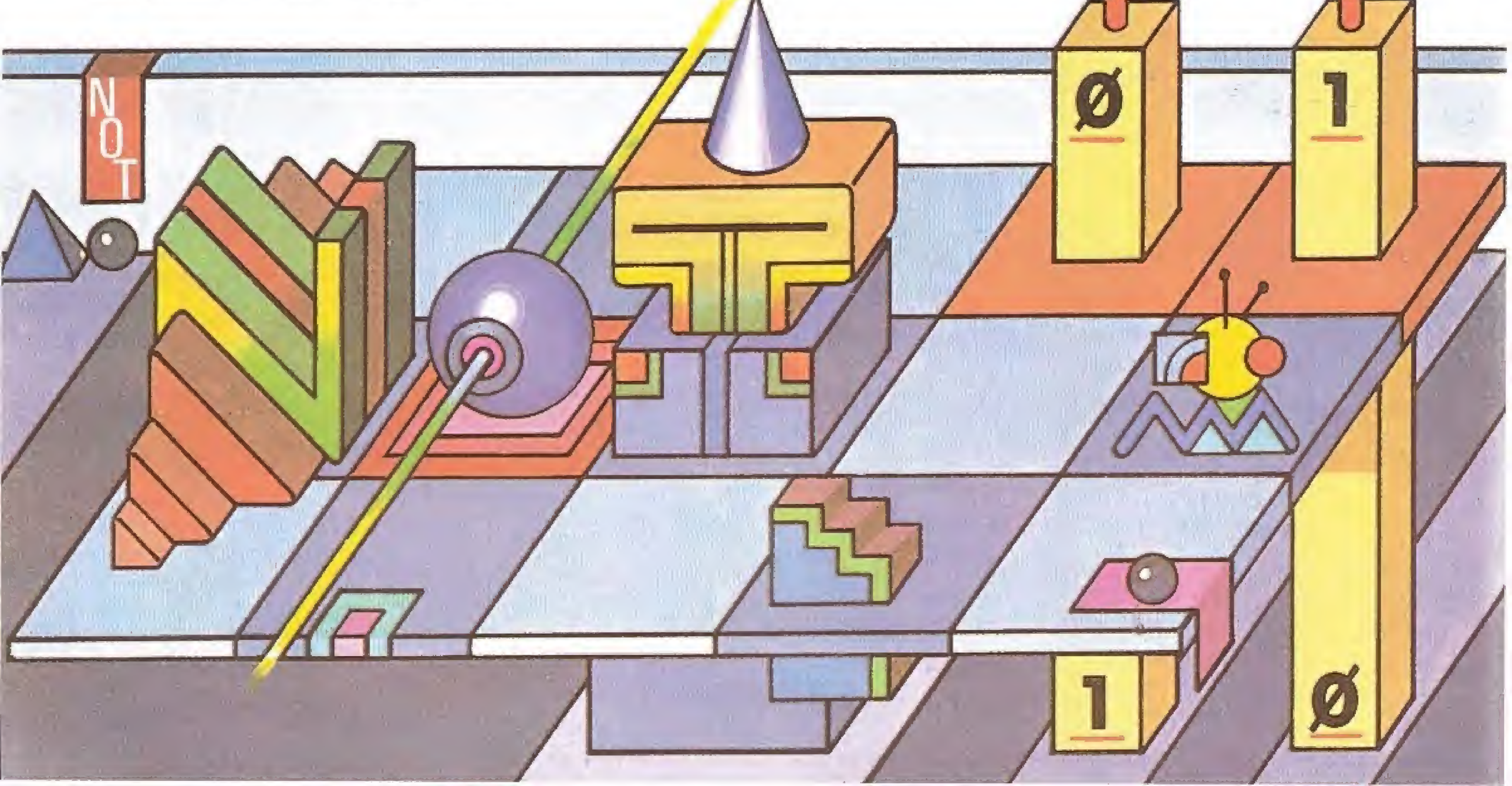
```
PRINT -1 OR 375
```

cuyo resultado es −1.

Veamos ahora una aplicación numérica para el operador NOT:

```
PRINT NOT 10.75,NOT -11
```

Los valores resultantes son −11 y 10. Así vemos que en efecto NOT suma 1 al número sobre el que se aplica y a continuación cambia su signo. Observa que el número en punto flotante se redondea por defecto al entero má próximo y que es posible estimar el valor real por medio de X = −(X + 1). De hecho este valor se convierte a un entero de cuatro bytes, todos cuyos bits quedan invertidos.



SORTEO PATROCINADO POR MICROUNO

****

La firma Microuno sortea 3 premios entre los lectores de INPUT Sinclair que enviéis una tarjeta postal conteniendo vuestros datos (nombre, apellidos, edad, dirección, ordenador...) y contando brevemente para que utilizáis el ordenador, si lo tenéis. La fecha tope de llegada es el próximo 15 de diciembre.

Los premios son:

Un ordenador Amstrad 6128.
Un dispositivo Wafadrive.
Doce programas de juego a elegir.

Escribir a:

INPUT Sinclair. (Sorteo Microuno).
Po. Castellana, 93-Planta 14.
28046 MADRID

DESENREDA TUS CADENAS

- COMPARACION Y ORDENACION DE CADENAS
- TROCEO
- USO DE CADENAS EN EL PROCESO DE TEXTOS

Las cadenas de caracteres se utilizan en toda clase de programas; de hecho se usan en todas las aplicaciones que requieran algo más que simples números. Aquí te presentamos unas cuantas maneras de sacar de ellas el máximo provecho.

Una cadena está constituída por una colección de caracteres. Puede tratarse de letras, números, signos de puntuación o cualquier otro de los símbolos del teclado. Además puedes ponerlas juntas en el orden que quieras.

Naturalmente lo normal es que una cadena contenga algún tipo de información útil. Por ejemplo, la cadena «PEDRO DIAZ 241061 S» contiene el nombre, el primer apellido,la fecha de nacimiento y el estado civil de una determinada persona. En total cuatro elementos de información. A su vez la fecha de nacimiento se puede subdividir en día, mes y año, por lo que en realidad hay un total de seis elementos de información.

Son muchas las ocasiones en las que suele ser necesario subdividir (trocear) una cadena de caracteres para extraer determinadas porciones de información, tal como ocurre con la fecha del ejemplo anterior. En otras ocasiones puede ser necesario sumar las cadenas. Puede que a veces también quieras medir la longitud de la cadena o calcular el valor de alguna de sus partes numéricas. Todo esto se puede hacer utilizando unas cuantas palabras clave del BASIC.

La suma de cadenas, la llamada **concatenación**, es la más fácil de estas operaciones. Para ejecutarla no tienes más que utilizar el símbolo +. Si por ejemplo A$ es igual a «HE» y B$ es igual a «AQUI», entonces A$ + B$ es igual a «HE AQUI». La concatenación es una operación que se limita a juntar unas cadenas de caracteres con otras; debe quedar claro que pone unas a continuación de otras, pero no suma sus valores numéricos. Así, «439» + «241» es igual a «439241» y no a 680.

COMPARACION DE CADENAS

El ordenador, además de poner unas cadenas junto a otras, es capaz de compararlas para ver si son iguales, como ocurre en este juego de adivinanzas:

```
10 LET G=1: GO TO INT (RND*
   6)*10+10
20 LET B$="MANZANA": GO TO
```

```
   80
30 LET B$="NARANJA": GO TO
   80
40 LET B$="PLATANO": GO TO
   80
50 LET B$="LIMON": GO TO 80
60 LET B$="MELON": GO TO 80
70 LET B$="FRESA": GO TO 80
80 CLS : PRINT "SOY UNA
   FRUTA,QUE FRUTA SOY?"
90 INPUT A$
100 IF A$=B$ THEN  GO TO 160
110 LET G=G+1
120 PRINT "FALLO"
130 FOR J=1 TO 200
140 NEXT J
150 GO TO 90
160 IF G=1 THEN  PRINT "HAS
    ACERTADO EN 1 INTENTO"
170 PRINT "HAS ACERTADO EN "
    ;G;" INTENTOS"
180 STOP
```

La línea 10 pone a 1 el contador de adivinanzas y a continuación lanza un dado electrónico para elegir una fruta. Las diferentes opciones están almacenadas en las líneas 20 a 70. Cualquiera que sea la línea a la que salte el ordenador, el nombre de la fruta correspondiente se almacena en B$, pasando después a la línea 80. Ahora tienes que acertar cuál es la fruta elegida e introducir tu conjetura mediante una sentencia INPUT con la variable A$. Esta variable se compara con A$ y si hay coincidencia total entre ambas, el ordenador presenta el mensaje «HAS ACERTADO A LA PRIMERA» o bien «HAS ACERTADO AL «,X,»INTENTO», como corresponda en cada caso. Si A$ no es igual a B$, el ordenador escribe «MAL» y tienes que ensayar de nuevo una respuesta. El programa sigue adelante hasta que la respuesta sea la correcta.

Fíjate bien en que aunque la respuesta elegida por tí sea la correcta, la condición A$ = B$ no se cumplirá a menos que la ortografía sea también correcta. Para que sea correcta, las dos cadenas de caracteres han de ser absolutamente idénticas, tanto las letras, como los números y los signos de puntuación.

Puedes utilizar esta técnica de comparación de cadenas de caracteres para comprobar las entradas introducidas desde el teclado, con una línea tal como la siguiente:

```
IF A$="SI" THEN PRINT "ESTAS
SEGURO?"
```

Observa que la condición anterior no se satisface si la palabra «sí» se teclea con letras minúsculas.

ORDENACION DE CADENAS

También puedes comparar las cadenas de caracteres utilizando los signos de desigualdad < y >. Podrías usarlos en una línea como la siguiente:

```
IF A$<B$ THEN PRINT "EL
PRIMERO ES ";A$
```

En este caso en la condición A$ < B$ se examina si la cadena A$ se sitúa antes o después que B$ cuando ambas se colocan en orden alfabético. Pero, ¡cuidado!, el ordenador hace la ordenación alfabética atendiendo al código ASCII de cada letra: a la A le corresponde el código ASCII 65 y a la Z el 90. El problema es que las letras minúsculas también tienen sus códigos ASCII: a la a le corresponde el 97 y a la z el 122. De esta forma, todas las cadenas de caracteres que empiezan por letras mayúsculas se sitúan por delante de las otras.

Y lo que es peor, los números, los signos de puntuación, los espacios y demás signos, también tienen sus códigos ASCII, por lo que el orden resultante para las cadenas puede quedar totalmente trastocado. No obstante, utilizando cuidadosamente los signos < y > sí se puede obtener una ordenación alfabética de las cadenas de caracteres.

TROCEADO DE CADENAS

Se puede extraer un carácter o un conjunto de caracteres del interior de una cadena. Para ello se utiliza una función A$(número TO número). A$ se emplea para identificar la cadena

que hay que fraccionar y los dos números indican el principio y el final del trozo.

Con la misma cadena A$ = «SR JUAN PEREZ», A$(1 TO 2) te da como resultado «SR», A$(4 TO 7) da «JUAN» y A$(9 TO 13) da «PEREZ». No siempre es necesario que especifiques ambos números. Si omites el primero, el ordenador da por supuesto que quieres empezar por el principio. Si por el contrario omites el último número, el ordenador supone que quieres llegar hasta el final.

El siguiente programa utiliza las funciones LEFT$, MID$ y LEN. Se trata de un juego de anagramas para dos personas. Una de ellas introduce una palabra que el ordenador seguida-

mente cifra y presenta en forma de anagrama para que la otra persona la adivine.

En la versión correspondiente al **ZX81**, tienes que expresar todas las variables con letras mayúsculas, suprimir las líneas 40 y 60 y los apóstrofos (') que siguen a las sentencias PRINT para fraccionar las líneas múltiples:

```
10 CLS : LET A$="": LET G=0
20 PRINT '"A N A G R A M A"
30 PRINT '"ESCRIBE LA
   PALABRA A TROCEAR"
40 POKE 23609,20: POKE 23658
   ,8: POKE 23624,63
50 INPUT W$
55 LET S$=W$
60 POKE 23624,56
70 CLS
80 FOR N=LEN W$ TO 1  STEP
   -1
90 LET M=INT (RND*N)+1
100 LET A$=A$+W$(M)
110 LET W$=W$( TO M-1)+W$(M
    +1 TO )
120 NEXT N
130 PRINT '"EL ANAGRAMA ES "
    ;A$
140 PRINT '"QUE PALABRA ES?"
160 INPUT  LINE G$
170 LET G=G+1
180 IF G$<>S$ THEN  PRINT '"
    FALLASTE, INTENTALO OTRA
    VEZ": GO TO 160
190 PRINT '"CORRECTO"
195 IF G=1 THEN  PRINT "HAS
    ACERTADO A LA PRIMERA":
    GO TO 210
200 PRINT "HAS NECESITADO ";
    G;" INTENTOS"
210 PRINT '"OTRA VEZ (SI/NO)
    ?"
220 LET A$=INKEY$: IF A$<>
    "S" AND A$<>"N" THEN GO
    TO 220
230 IF A$="S" THEN  RUN
```

Cuando ejecutes este juego, observarás que la primera palabra que tecleas no aparece en la pantalla. Lo hemos programado así para que tu oponente no pueda saber cuál es. Cada ordenador utiliza un método diferente para conseguir esto. El **Spectrum** lo que hace es escribir la palabra con el color de fondo de pantalla, de manera que resulta invisible.

La rutina de cifrado está contenida en las líneas 80 a 120. Esta rutina extrae caracteres de la pantalla de forma aleatoria y los pone todos juntos en A$, que se convierte así en un anagrama.

En la línea 90 se elige un número aleatorio M comprendido entre 1 y la longitud de la palabra, y a continuación la línea 100 extrae la letra M-sima y se la añade a A$. La línea 110 borra este carácter de la palabra original, suprimiendo su parte izquierda hasta el número M y añadiendo la parte restante a la derecha de M. La palabra resultante tiene ahora un carácter menos, pero la vez siguiente, la variable N también es un carácter menor por lo que el número aleatorio M está otra vez restringido a la longitud de palabra.

Llegará un momento en que se haya extraído todos los caracteres, quedando el anagrama impreso en la pantalla, momento en que el programa le preguntará a tu oponente cuál era la palabra original. Cuando teclee con la sentencia INPUT, la palabra correcta el programa responde cuántos intentos se han necesitado y te brinda una nueva oportunidad.

Sería muy sencillo modificar este programa para que se puedan leer las palabras a través de una serie de sentencias DATA en vez de utilizar separadamente un INPUT cada vez. También puedes incluir un sistema de puntuaciones que, por ejemplo, asigne diez puntos cada vez que se acierte a la primera, nueve puntos a la segunda y así sucesivamente.

Otro posible uso de este método de troceo es la manipulación de fechas. Incluso en los casos en que las fechas se introducen en forma de números, por ejemplo 27/03/51, en vez del 27 de marzo de 1951. Después de todo, puedes manipular una fecha expresada de esta forma usando las leyes matemáticas. También puedes procesar diferentes partes de una fecha. Puedes calcular por ejemplo la edad de una determinada persona tecleando su fecha de nacimiento, o puedes calcular el número de días transcurridos entre dos fechas dadas; no tienes más que manejar separadamente la parte de los años, la de los meses y la de los días.

Las funciones de selección con que cuenta el **Spectrum** para quedarse con la parte derecha, izquierda o central de una cadena de caracteres, te permiten separar con facilidad los días, los meses y los años de una determinada fecha. Si ahora utilizas las funciones VAL, obtendrás a partir de la cadena de caracteres resultante un número con el que puedes hacer sumas, restas multiplicaciones y divisiones.

LONGITUD DE LAS CADENAS

A veces interesa conocer la longitud de una cadena. Si por ejemplo sólo dispones de una limitada capacidad de memoria para cada elemento de información introducido desde el teclado, o sólo un espacio limitado de pantalla para presentar dicha información, puede ser de gran ayuda comprobar la longitud de la cadena antes de seguir con el programa.

La función LEN(A$) te da como resultado el número de caracteres que contiene la cadena A$. Se trata de una función numérica y no de una cadena de caracteres, por lo que puedes manipularla con arreglo a las leyes normales del álgebra. Por ejemplo, si A$ = «Sr Juan Perez», LEN(A$) = 13. Pero supongamos que te piden que teclees un nombre para un determinado fichero cuyo formato tabulado para representación en pantalla sólo tiene sitio para 11 letras, por lo que puede ser que haya que enviar un mensaje para que se recorte la entrada, tal como el siguiente:

```
10 PRINT "ESCRIBE EL NOMBRE"
20 INPUT A$
30 IF LEN(A$)>11 THEN PRINT
   "LO SIENTO SOLO CABEN 11
   CARACTERES": GO TO 10
```

El usuario podría así recortar su entrada y teclear únicamente «JUAN PEREZ».

En otros casos podría resultar más fácil truncar automáticamente la entrada. Esto se hace con una línea tal como la siguiente:

```
IF LEN(A$)>15 THEN LET A$=
A$(TO 15)
```

CONVERSION DE CADENAS A NUMEROS

Uno de los usos de las variables de cadenas de caracteres es conseguir que las entradas introducidas por el teclado sean correctas. Por ejemplo, si escribes un programa que contenga un par de líneas como las siguientes:

```
100 PRINT "ESCRIBE UN
    NUMERO"
110 INPUT A
```

el ordenador se quedará esperando que teclees algún número. Si por error tecleas algún carácter no numérico, casi todos los ordenadores interrumpirán la representación en pantalla y enviarán un mensaje estándar de error. Dicho mensaje te informará de que hay algo que está mal pero no te dirá lo que es.

Pero si en lugar de esto utilizas:

```
110 INPUT A$
```

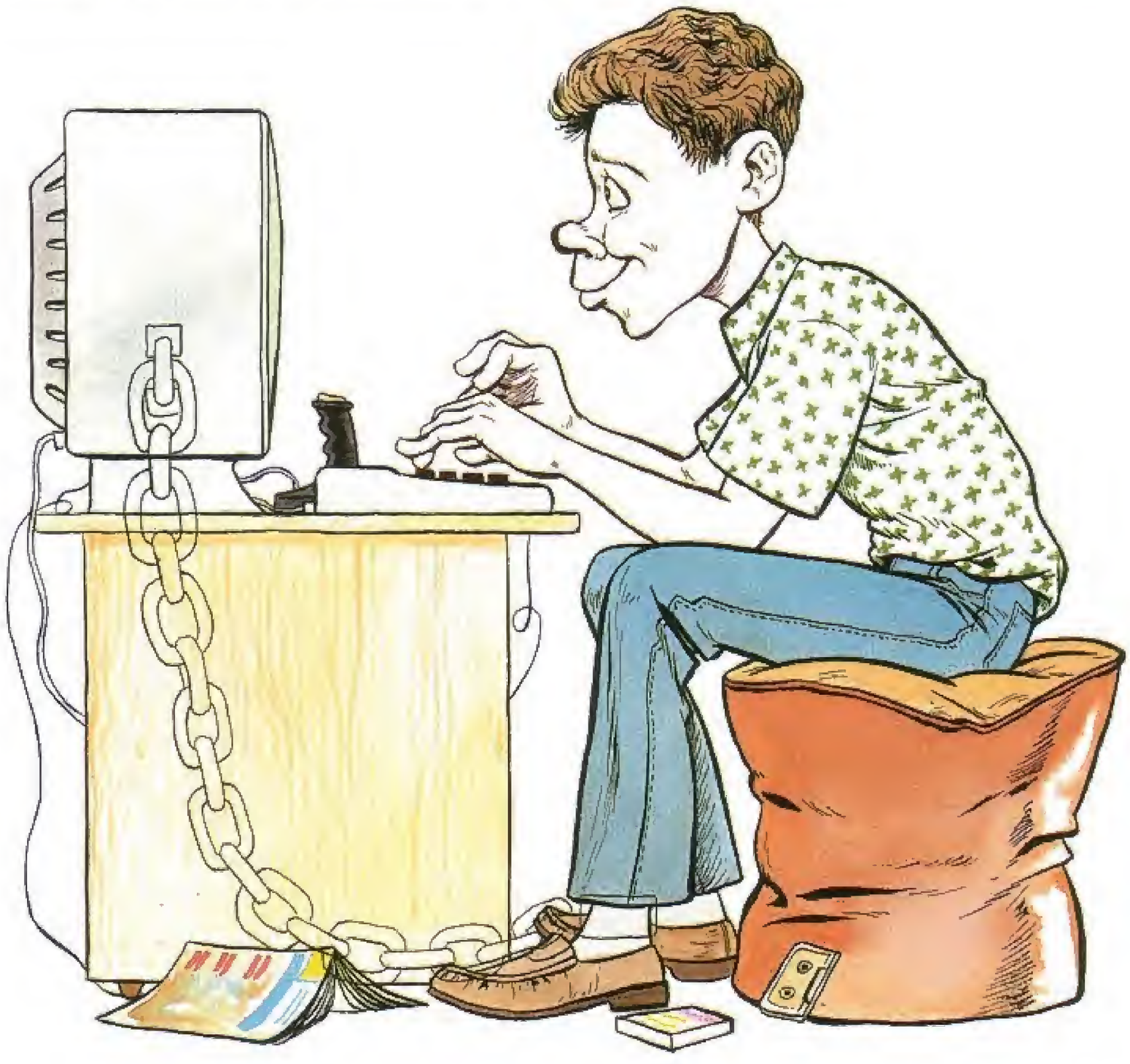
en este caso el usuario puede teclear casi cualquier cosa y el ordenador lo aceptará. El siguiente paso es convertir la cadena de caracteres en un número. Para hacerlo, utiliza la función VAL(A$).

Por desgracia esto no siempre te funcionará en el **Spectrum**. Si intentas obtener el valor de una cadena de caracteres, recibirás un mensaje de error. Sólo se puede utilizar VAL en el **Spectrum** cuando la cadena de caracteres está constituída íntegramente por números. Así, con VAL(«1984») se obtiene 1984, lo cual es correcto, pero VAL(«26/10/84) te dará 0.03095, debido a que el **Spectrum** utiliza VAL para evaluar la expresión 26/10/84.

CONVERSION DE NUMEROS A CADENAS

La función STR$ tiene un resultado casi opuesto al de VAL. Sirve para transformar una cadena en un número. La ventaja de esto es que las cadenas de caracteres pueden ser manipuladas de forma distinta a los números, por ejemplo con ayuda del troceo y la concatenación que ya hemos visto, por lo que STR$ tiene varias aplicaciones.

El siguiente programa sirve para convertir en binario un número decimal. Aunque los ordenadores mane-



ORDENA TUS CADENAS

Cuando el ordenador extrae de manera casi mágica las direcciones de centenares de entradas separadas y las presenta en la pantalla, es fácil imaginar que la máquina es más inteligente de lo que realmente es. Sucede casi como si el ordenador estuviera leyendo directamente las entradas y a continuación decidiendo lo que tiene que hacer.

No te confundas. El ordenador se limita a ir extrayendo una parte de la cadena a medida que recibe la información. Si resulta que una parte de esas cadenas contiene un galimatías, pues eso es lo que te encontrarás.

Para asegurarte de que obtienes la misma información de cada cadena, tienes que garantizar que la información se almacena siempre en el mismo sitio. Los investigadores profesionales con frecuencia utilizan tarjetas de entrada con un formato normalizado, que constituyen una de las ayudas más valiosas que te puedes procurar. Para ello no necesitas más que una serie de espacios que contengan los caracteres, rotuladas con arreglo al lugar donde debe empezar y acabar cada elemento de información.

jen toda su aritmética interna en binario, el lenguaje BASIC sólo puede manejar números decimales y, en algunos casos, hexadecimales. Por ello, cuando en un programa en BASIC se presenta un número en binario, hay que manejarlo como una cadena de caracteres.

```
10 PRINT "DECIMAL A BINARIO"
20 PRINT "ESCRIBE EL NUMERO
   DECIMAL"
30 INPUT D
40 LET B$=""
50 LET B$=STR$ (D-INT (D/2)*
   2)+B$
60 LET D=INT (D/2)
70 IF D<>0 THEN  GO TO 50
80 PRINT "EL NUMERO BINARIO
   ES ";B$
```

Al introducir un número decimal positivo, la línea 40 hace la cadena B$ igual a la cadena nula, que después va llenando progresivamente con dígitos a medida que progresa el cálculo en el bucle de las líneas 50 a 80.

La línea 50 es la que realmente se encarga de construir el número binario. Para ello resta dos veces la parte entera de la mitad del número decimal del propio número decimal. Esta es una forma sencilla de comprobar si un número es par o impar. Si es impar, el resultado es 1, mientras que si es par el resultado es 0. Naturalmente, estos son los dígitos binarios buscados.

PROCESAMIENTO DE TEXTOS

Las funciones de manejo de cadenas de caracteres se utilizan ampliamente, en especial en proceso de textos. Una necesidad que se plantea con frecuencia es la sustitución de una palabra por otra en toda la extensión de un documento (posiblemente debido a que has descubierto una falta de ortografía). Naturalmente, si la nueva palabra tiene una longitud diferente de la original, tienes que desplazar el resto del texto para dejarle sitio. El siguiente programa te muestra la manera de hacerlo:

```
10 INPUT "ESCRIBE EL TEXTO "
   ; LINE T$: PRINT T$
20 INPUT "PALABRA A
   REEMPLAZAR "; LINE W$:
   LET W=LEN W$
30 INPUT "NUEVA PALABRA ";
   LINE N$: LET N=LEN N$
35 LET P=0
40 LET P=P+1
50 IF P+W-1>LEN T$ THEN GO
   TO 100
60 IF T$(P TO P+W-1)<>W$
   THEN  GO TO 40
70 LET T$=T$( TO P-1)+N$+T$
   (P+W TO ): GO TO 40
100 PRINT T$
110 GO TO 20
```

Empieza introduciendo una frase más bien sencilla y a continuación ensaya el efecto de sustituir algunas de sus letras o palabras. En las palabras cortas tales como «a» o «un» conviene teclear además un espacio antes y después para evitar que se apliquen también los cambios cada vez que se presenten dichas palabras embebidas en otras más largas, tales como «para» o «unidad».

El programa funciona de la siguiente manera: La línea 40 envía a P a empezar la fase de búsqueda al principio del texto. Las líneas 50 y 60 detectan la primera aparición de la palabra o grupo de letras que pretendes sustituir y a continuación la línea 70 realiza la sustitución. Lo que hace es tomar el texto original, remplazar la palabra antigua por la nueva y añadir el texto restante. El proceso se repite hasta que se ha realizado la sustitución en todos los sitios en que aparece la palabra que se quiere sustituir, imprimiéndose a continuación en la línea 100.

Como has visto, se trata de un ejemplo muy simple, los procesadores del mundo real son mucho más complicados. Sin embargo puede servir para darte una idea de algunas aplicaciones prácticas de lo que se puede hacer con las cadenas de caracteres.

SOLUCIONES «INGENIOSAS»

La mecánica no existe sólo para los ingenieros. Es el estudio de las interacciones entre las fuerzas e interviene en todo tipo de actividades cotidianas, susceptibles de ser analizadas por tu ordenador.

La construcción de muchos de los mayores monumentos del mundo, tales como las pirámides de Egipto o el complejo megalítico de Stonehenge, resulta especialmente sorprendente por el hecho de que fueron construídos mucho tiempo antes del desarrollo de los dispositivos tecnológicos considerados como necesarios para dichas tareas.

Actualmente serían muy pocos los constructores que se atrevieran a abordar este tipo de trabajos sin disponer de la maquinaria adecuada para cortar y transportar las enormes piedras o para elevar las grandes masas de material a grandes alturas sobre el suelo. Y sin embargo hasta la más sofisticada de dichas máquinas está basada en unos principios fundamentales que ya eran conocidos y utilizados por los ingenieros primitivos. Entre todos ellos destaca especialmente el principio de la mecánica, que es la ciencia que se ocupa del estudio de las fuerzas.

Ya se trate de un sistema estacionario (como un edificio) o en movimiento (como los elementos de una máquina), siempre hay fuerzas que actúan sobre él. Se pueden valorar estas fuerzas de un modo instintivo o a partir de la experiencia, igual que hacían los constructores primitivos, y lo que hacemos todos en las actividades de la vida diaria. Por ejemplo cuando vas a levantar con una mano una taza de café, automáticamente aplicas la fuerza correcta para levantarla sin que el café salga volando. Y si vas a colocar una estantería, tienes una idea bastante precisa del tipo de maderas que tienes que utilizar y de cómo colocar los soportes para evitar que se comben por acción del peso sobre la misma.

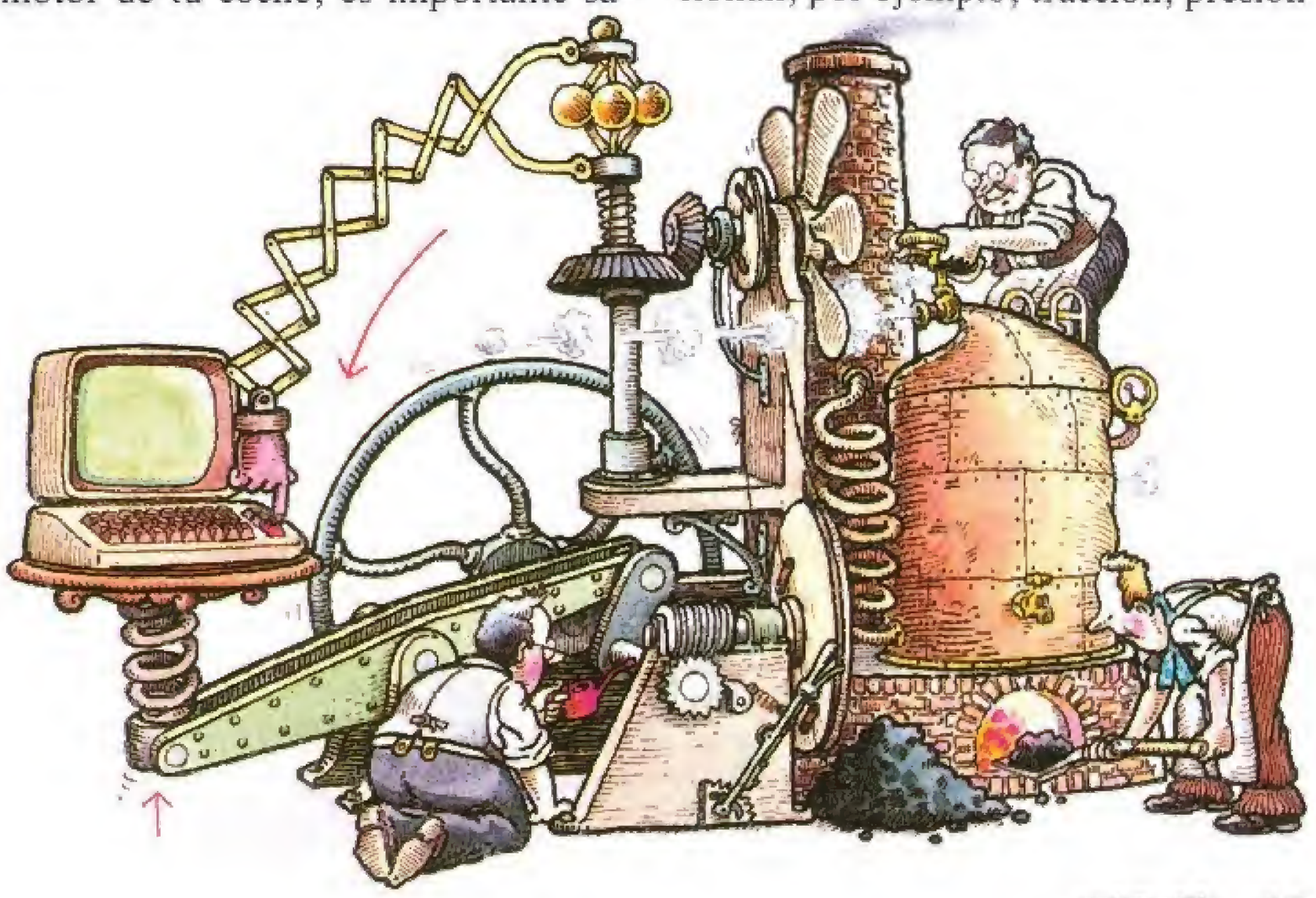
La estimación de fuerzas de esta manera está muy bien para las aplicaciones corrientes, pero hay muchas ocasiones en que se requiere una mayor precisión. En este tipo de análisis es donde intervienen los ordenadores.

Naturalmente, puedes utilizar estimaciones basadas en la experiencia para programar simulaciones en ordenador de sistemas en los que intervienen fuerzas. En su nivel más sencillo, esto es lo que se hace en el tipo de simulación empleado en muchos programas de juegos. Así puedes estimar por ejemplo lo lejos que caerá una espada cuando es arrancada de la mano de un enemigo, o con qué fuerza se escuchará el ruido producido por el choque de dos objetos.

Pero cuando utilices como herramienta el ordenador, es seguro que necesitarás mayor precisión. Por ejemplo, tanto si eres un ingeniero que está proyectando un puente como un mecánico aficionado intentando construir un artilugio para levantar el motor de tu coche, es importante saber cuándo se alcanzará el punto de ruptura de la estructura. Si utilizas tu ordenador para analizar las fuerzas que actúan sobre una estructura en condiciones de carga variable, necesitas una precisión absoluta, así como un buen entendimiento de cómo calcular las fuerzas.

LA MAS SENCILLA DE LAS MAQUINAS

A medida que se construyen estructuras más pesadas, el problema de moverlas se va convirtiendo cada vez en un reto mayor. Durante la década de los sesenta, el poderoso cohete **Saturno V** que sirvió para enviar hombres a la Luna, fué transportado sobre la mayor plataforma rodante del mundo. En la actualidad, las plataformas petrolíferas del Mar del Norte, apoyadas sobre colchones neumáticos tipo *hovercraft* o plásticos lubricantes, son remolcadas hasta la playa y botadas al mar. Estas técnicas y otras parecidas dependen para su realización de alguna clase de máquinas, que les proporcionan, por ejemplo, tracción, presión



o elevación. A su vez estas máquinas son una superposición de unos cuantos dispositivos básicos que han sido utilizados durante siglos.

La más primitiva de estas máquinas es la palanca. En su forma más sencilla, la palanca es una barra rígida, uno de cuyos extremos se coloca debajo de un objeto mientras que se actúa sobre el otro extremo para mover el objeto. Con este simple artificio, una persona puede mover una carga varias veces superior a su propio peso. De hecho, se atribuye a Arquímedes, el gran científico de la Antigüedad, la frase «*Dadme un punto de apoyo y moveré el mundo*» en una clara referencia a la ley de la palanca. Según afirmaba, sería capaz de mover la Tierra siempre que tuviera una palanca suficientemente larga y un buen fulcro. El fulcro o punto de apoyo es el punto alrededor del cual pivota la palanca, por lo que en el ejemplo anterior es simplemente un lugar sobre el suelo en el que se apoya la barra rígida.

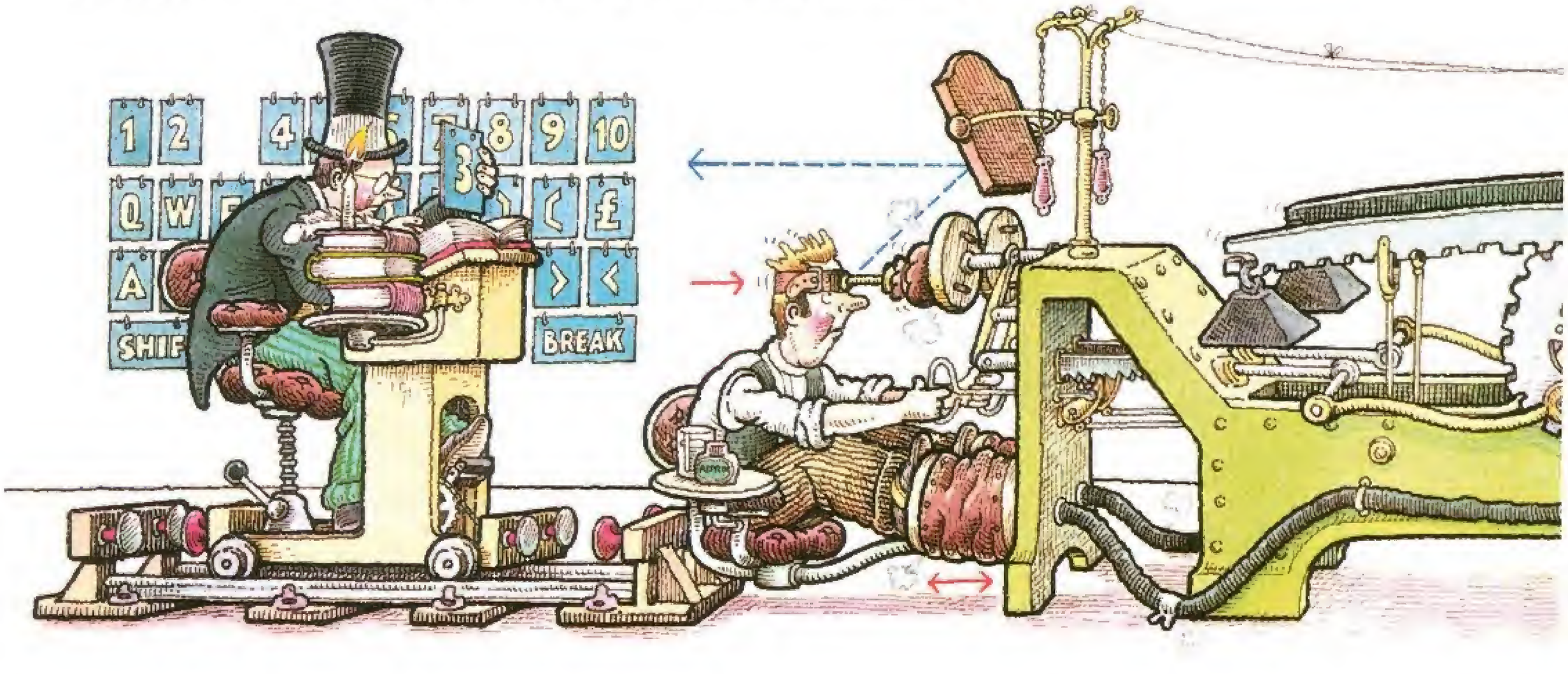
La disposición más normal de la palanca tiene el fulcro no en uno de los extremos de la barra rígida, sino en algún punto entre ambos extremos. Teclea el primer programa que figura a continuación y tendrás una demostración de esto.

Teclea para Spectrum

```
30 BORDER 0: PAPER 0: INK 7
   : CLS : OVER 1
40 INPUT "DISTANCIA DESDE
   LA IZQUIERDA AL PUNTO
   DE APOYO "'"[8*ESPACIO]
   (1-8 MTRS)";D
50 IF D<1 OR D>8 THEN GO TO
   40
60 LET W=(10-D)/D
70 PRINT AT 1,0#"PESO
   REQUERIDO PARA
   EQUILIBRAR  100 KG=";W*
   100;"KG"
100 PLOT  INK 4;0,15: DRAW
    INK 4;255,0: FOR N=0
    TO 55: PLOT (28+20*D)-N
    /3,71-N
110 DRAW  INK 7;2*((28+20*D
    )-PEEK 23677),0: NEXT N
120 LET A=D-10: LET A=ATN
    (A/(1-A*A))+2*ATN (1)
130 FOR B=A TO 2*ATN (1)
    STEP (2*ATN (1)-A)/10
140 FOR K=1 TO 2: GO SUB
    1000: GO SUB 1500
160 NEXT K
170 NEXT B
180 LET B=2*ATN (1): GO SUB
    1000: GO SUB 1500
190 IF INKEY$="" THEN GO TO
    190
200 RUN
1000 PLOT 128-100*SIN (B),
     71-20*D*COS (B)
1011 DRAW 127+100*SIN (B)-
     PEEK 23677,71+(200-20
     *D)*COS (B)-PEEK 23678
1025 DRAW 0,-8: DRAW -10,0:
     DRAW 0,-10: DRAW 20,0:
     DRAW 0,10: DRAW -9,0:
     POKE 23678,(PEEK 23678
     )+6
1030 RETURN
1500 PLOT 128-100*SIN (B),
     70-20*D*COS (B)
1510 LET E=SQR (SQR W)
1520 DRAW 0,-8: DRAW -10*E,
     0: DRAW 0,-10*E: DRAW
     20*E,0: DRAW 0,10*E:
     DRAW -9*E,0
1530 RETURN
```

Al ejecutar el programa aparecerá un mensaje en la pantalla (línea 40) para que especifiques la distancia (D) desde un punto de apoyo hasta el extremo izquierdo de una barra de 10 metros. La línea 60 calcula entonces el peso (W) requerido en dicho extremo para equilibrar una carga de 100 Kg colocada en el otro extremo; la línea 70 presenta en la pantalla el valor de W. Las líneas 100 a 110 se ocupan de dibujar el fulcro. La línea 120 define las variables correspondientes al ángulo de la palanca y las líneas 130 a 170 animan el equilibrio de la barra mediante la llamada a una rutina (líneas 1000 a 1030) para dibujar la barra y otra (líneas 1500 a 1530) para dibujar los pesos. La línea 180 dibuja la posición final de la barra y los pesos.

Prueba a introducir diferentes valores en D cada vez que ejecutes el programa, y observa cómo se requiere un mayor esfuerzo (es decir un valor mayor de W) a medida que el fulcro va estando más alejado de la carga de 100 Kg. Por el contrario, a medida que el fulcro se va colocando más cerca de la carga, la palanca tiene un efecto mucho mayor, por lo que hace falta mucho menor esfuerzo.



VENTAJA MECANICA

Existe una fórmula matemática muy sencilla para calcular las longitudes y los pesos de un montaje de este tipo, la cual da muy buenos resultados al considerar el efecto de carga sobre cualquier tipo de barra o viga, en el supuesto de que conozcas la posición del punto sobre el que actúa la carga. Dicha fórmula establece que el esfuerzo o potencia multiplicado por la distancia desde el punto donde se aplica hasta el fulcro, es igual a la carga o resistencia multiplicada por su distancia al fulcro. Escribiendo esto con variables podemos poner:

$$E \times DE = C \times DC$$

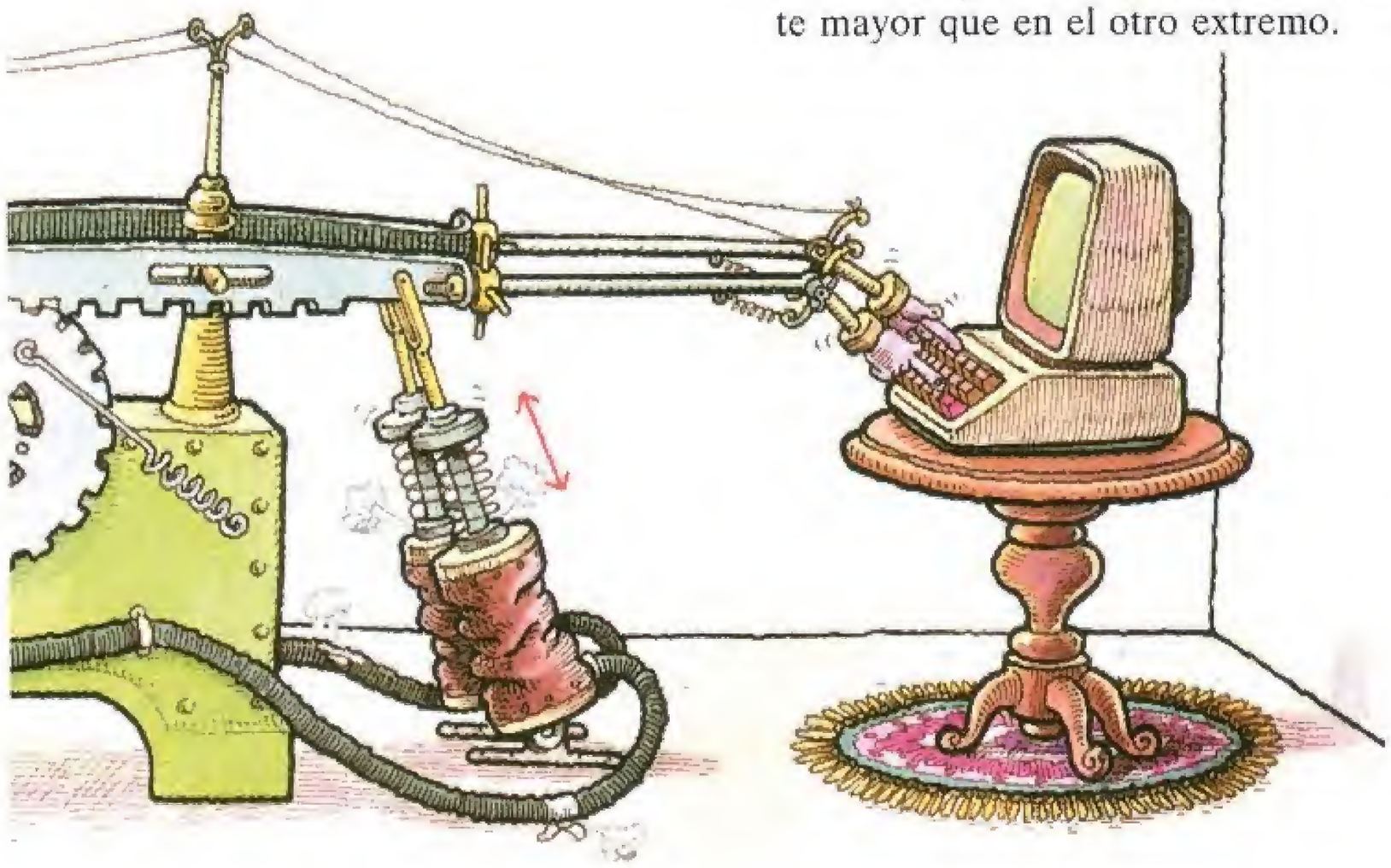
En el programa anterior se utilizó W en lugar de E; la longitud total de la barra es 10 , por lo que DC es igual a 10 − DE. Si el valor de C es igual a 1, la fórmula se convierte en

$$W = (10 - DE)/DE$$

que es la forma en que se utiliza en la línea 60 del programa anterior. Se te pide que especifiques DE (D en el programa), para poder calcular el esfuerzo o potencia a aplicar. Sea cual sea la forma en que utilices la fórmula, siempre podrás calcular la variable que falte si conoces las demás.

Este hecho tan simple se utiliza con gran provecho en las balanzas ordinarias. Como sabes, se trata de una viga de metal que apoya y pivota alrededor de un punto de su centro. Esto significa que DE = DC, ecuación que se puede simplificar para dar E = C. Ahora, para pesar un objeto cualquiera, no tienes más que colocarlo en uno de los platillos que cuelgan de los brazos y poner pesos conocidos en el otro platillo hasta que la balanza quede equilibrada. Esto está muy bien cuando se trata de pesar pequeñas cantidades, como puede ocurrir en una frutería, pero ¿qué sucede cuando hay que pesar un vehículo comercial tal como un camión cargado de mineral o de carbón? De hecho se aplica el mismo principio, pero en este caso en vez de situar el punto de apoyo en el centro se coloca muy cerca de la carga, por lo que se puede seguir equilibrando la balanza con pesos pequeños.

Cuando ya hayas jugado bastante con este programa, y hayas comprobado de qué forma el alargamiento de la palanca disminuye el esfuerzo necesario, estarás en condiciones de descubrir palancas en casi cualquier máquina. Esta disminución del esfuerzo se suele llamar ventaja mecánica y está dada por el cociente entre el esfuerzo y la carga. Si transformas adecuadamente la fórmula anterior, encontrarás que la ventaja mecánica es igual a la distancia que se desplaza el esfuerzo dividida por la distancia que se desplaza la carga. De modo que la próxima vez que tires de una palanca —que puede ser el picaporte de una puerta o un freno de bicicleta— observa cómo el desplazamiento del extremo donde se aplica el esfuerzo es bastante mayor que en el otro extremo.

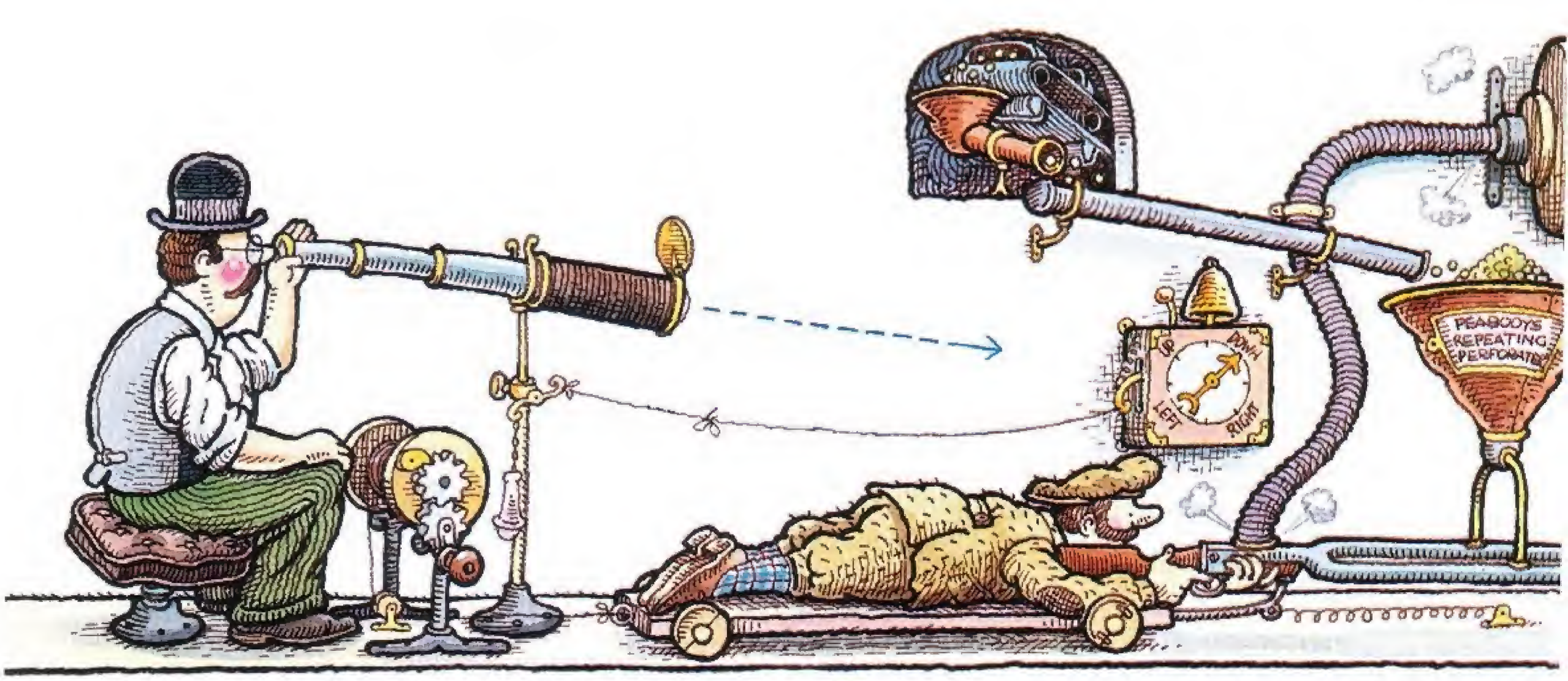


ALCANZANDO MAYORES ALTURAS

Para empujar y tirar de cosas, lo mejor es utilizar palancas, pero cuando lo que hace falta es elevar una determinada carga a una gran altura, la palanca ha de ser modificada. Este es el objeto de los sistemas de poleas, que también te permiten obtener ventaja mecánica. Para ver demostrado esto, teclea y ejecuta el siguiente programa:

Teclea para Spectrum

```
10 CLEAR 32399: RESTORE :
   GO SUB 510: BORDER 0:
   PAPER 0: INK 7: CLS
20 PRINT AT 10,0;"CUANTAS
   POLEAS (2,4 0 6)?"
30 LET A$=INKEY$: IF A$<>
   "2" AND A$<>"4" AND A$<>
   "6" THEN  GO TO 30
40 PRINT TAB (10);A$: LET
   NP=VAL (A$): LET L=25*NP
   -50
45 IF NP=2 THEN  POKE 32431
   ,25
46 IF NP=4 THEN  POKE 32431
   ,17
47 IF NP=6 THEN  POKE 32431
   ,14
50  PRINT : PRINT "ESFUERZO
   REQUERIDO=";INT (1000/
   NP): PRINT "KILOS PARA
   ELEVAR 1 TONELADA"
55 FOR M=1 TO 500: NEXT M
60 CLS : GO SUB 1000
70 LET SP=120
90 FOR K=1 TO 50
100 RANDOMIZE USR 32400
105 PLOT  OVER 1;191-L-NP,
    SP: DRAW  OVER 1;8,0
110 PLOT  OVER 1;191-L-NP,
    SP: DRAW  OVER 1;8,0
120 LET SP=SP-NP: IF SP<=0
    THEN  LET SP=120
150 NEXT K
160 IF INKEY$="" THEN  GO
    TO 160
170 RUN
510 FOR N=32400 TO 32491
520 READ A: POKE N,A
```



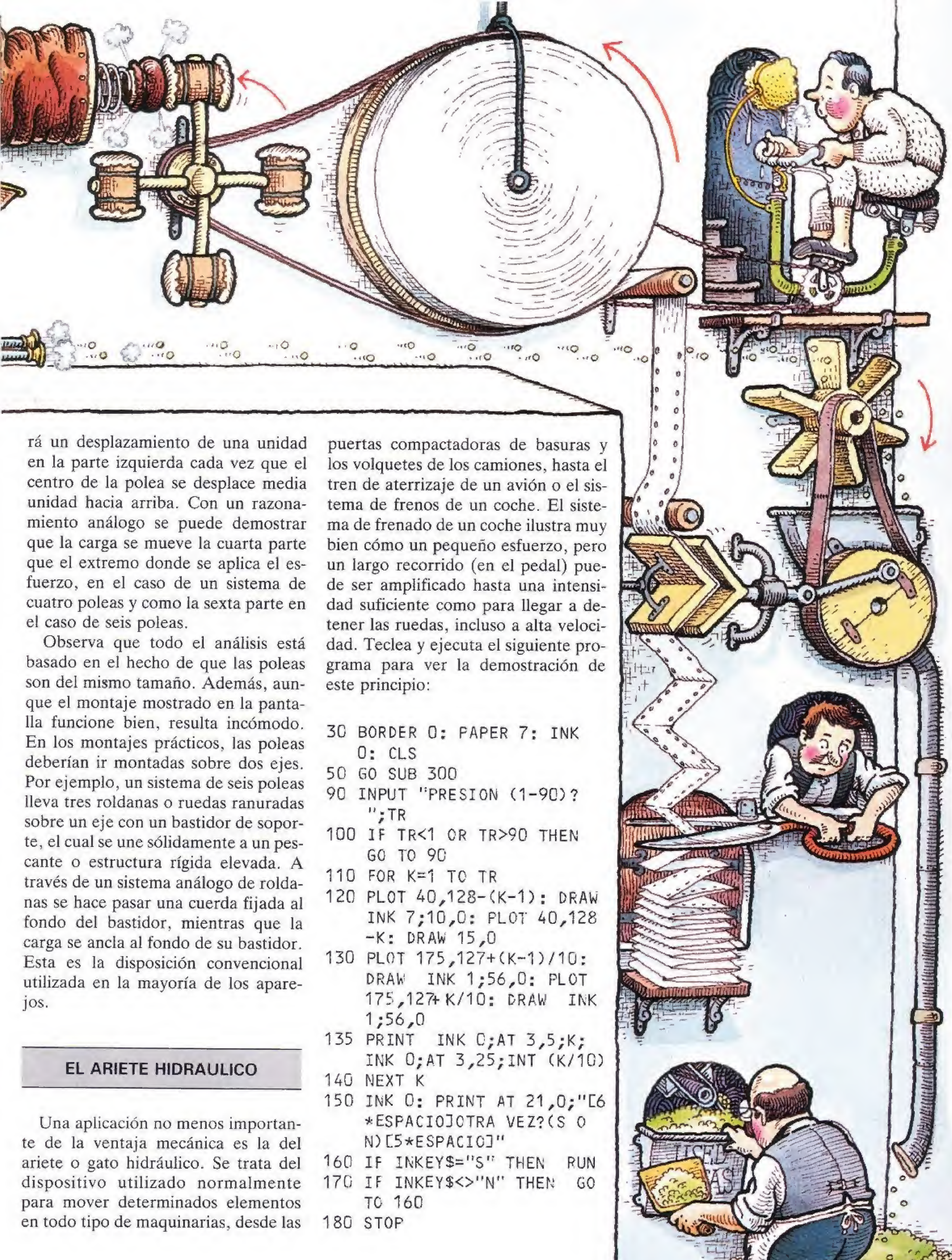

```
530 NEXT N
540 DATA 62,60,79,230,192,
    15,15,15,198,64,103,121
    ,230,7,132,103,121,135,
    135,230,224,111,62
550 DATA 175,254,192,208,145
    ,216,8,14,14,125,177,111
    ,62,30,254,32,208,145,
    216,60,79,6,0,197,229,17
    ,224,91,237,176,225
560 DATA 193,217,8,167,40,30
    ,71,217,124,60,87,93,230
    ,7,32,10,123,198,32,95,
    56,4,122,214,8,87,235
570 DATA 229,197,237,176,193
    ,225,217,16,227,217,201
580 RETURN
1000 PLOT 0,170: DRAW 255,0
1010 FOR K=1 TO NP STEP 2
1020 CIRCLE (232-K*26),140
     ,13
1030 CIRCLE (258-K*26),50,13
1040 NEXT K
1050 PLOT 245,170: DRAW 0,
     -125
1060 FOR K=1 TO NP-1
1070 PLOT 246-K*26,140: DRAW
     0,-90
1080 NEXT K
1090 PLOT 197-L-NP,140: DRAW
     0,-140
1100 PLOT 208,141: DRAW (256
     -NP*26)-PEEK 23677,0
1110 PLOT 234,50: DRAW (282-
     NP*26)-PEEK 23677,0
1120 PLOT 206,170: DRAW 0,-
     28: PLOT 258-NP*26,170:
     DRAW 0,-28
1130 PLOT 231-L*2/3,50: DRAW
     0,-20
1140 PLOT 232-L/3,50: DRAW 0
     ,-20
1145 DRAW (231-L*2/3)-PEEK
     23677,0
1150 PLOT 231-L/2,29: DRAW
     0,-10
1160 DRAW -9,0: DRAW 0,-10:
     DRAW 19,0: DRAW 0,10:
     DRAW -9,0
1170 PLOT 180-L,0: DRAW
     (180-L-20/SQR (NP))-
     PEEK 23677,0: DRAW 0,
     (20/SQR (NP))-PEEK
     23678: DRAW (180-L)-
     PEEK 23677,0: DRAW 0,
     0-PEEK 23678
1210 RETURN
```

El programa te envía un mensaje para que especifiques el número de poleas que quieres que intervenga en la demostración (línea 20). Para empezar, teclea un 2 y aparecerá en la pantalla (línea 50) el esfuerzo necesario para elevar una carga de 1000 Kg (una tonelada). Seguidamente el programa realiza una animación de la carga cuando está siendo elevada. La rutina que se extiende entre las líneas 1000 y 1210 (a la cual se llama desde la línea 60) sirve para dibujar el sistema de cuerdas, polea fija y apoyos, mientras que las líneas 90 a 150 se ocupan de la polea móvil, la carga y la cuerda que se mueve.

Ejecuta nuevamente el programa pero introduciendo ahora 4 y después 6 como respuesta al mensaje inicial y compara los valores del esfuerzo y de la distancia resultantes para el extremo libre de la cuerda. Como el esfuerzo es igual a la carga dividida por el número de poleas, el esfuerzo resultante será igual a 1/2, 1/4 o 1/6 de los 1000 Kg para los casos de 2, 4 o 6 poleas. En consecuencia, cuanto mayor sea el número de poleas empleadas, menor será el esfuerzo requerido para levantar la misma carga. Al igual que en el caso de la palanca, la ventaja mecánica está dada por la relación carga/esfuerzo. Para el caso de dos poleas esta relación es 1000 dividido por 500, lo que supone una ventaja igual a 2. Análogamente, la ventaja mecánica es 4 para el caso de cuatro poleas y 6 cuando se utilizan seis poleas.

AHORRANDO ESFUERZOS

Igual que en el caso de la palanca, sigue valiendo en este caso la relación entre la distancia que se desplaza la carga y la que se desplaza el esfuerzo. Aunque en el caso de dos poleas, por ejemplo, la animación revela que el extremo donde se aplica el esfuerzo se desplaza el doble de la distancia recorrida por la carga, esto no resulta muy fácil de verificar. Si consideras que la cuerda del extremo izquierdo de la polea fija se desplaza hacia abajo una unidad de longitud, entonces la cuerda del extremo derecho se desplaza dos unidades hacia arriba. La diferencia con respecto al caso de una única polea fija es que la cuerda del lado derecho está fija, por lo que se produci-



rá un desplazamiento de una unidad en la parte izquierda cada vez que el centro de la polea se desplace media unidad hacia arriba. Con un razonamiento análogo se puede demostrar que la carga se mueve la cuarta parte que el extremo donde se aplica el esfuerzo, en el caso de un sistema de cuatro poleas y como la sexta parte en el caso de seis poleas.

Observa que todo el análisis está basado en el hecho de que las poleas son del mismo tamaño. Además, aunque el montaje mostrado en la pantalla funcione bien, resulta incómodo. En los montajes prácticos, las poleas deberían ir montadas sobre dos ejes. Por ejemplo, un sistema de seis poleas lleva tres roldanas o ruedas ranuradas sobre un eje con un bastidor de soporte, el cual se une sólidamente a un pescante o estructura rígida elevada. A través de un sistema análogo de roldanas se hace pasar una cuerda fijada al fondo del bastidor, mientras que la carga se ancla al fondo de su bastidor. Esta es la disposición convencional utilizada en la mayoría de los aparejos.

EL ARIETE HIDRAULICO

Una aplicación no menos importante de la ventaja mecánica es la del ariete o gato hidráulico. Se trata del dispositivo utilizado normalmente para mover determinados elementos en todo tipo de maquinarias, desde las puertas compactadoras de basuras y los volquetes de los camiones, hasta el tren de aterrizaje de un avión o el sistema de frenos de un coche. El sistema de frenado de un coche ilustra muy bien cómo un pequeño esfuerzo, pero un largo recorrido (en el pedal) puede ser amplificado hasta una intensidad suficiente como para llegar a detener las ruedas, incluso a alta velocidad. Teclea y ejecuta el siguiente programa para ver la demostración de este principio:

```
30 BORDER 0: PAPER 7: INK
   0: CLS
50 GO SUB 300
90 INPUT "PRESION (1-90)?
   ";TR
100 IF TR<1 OR TR>90 THEN
    GO TO 90
110 FOR K=1 TO TR
120 PLOT 40,128-(K-1): DRAW
    INK 7;10,0: PLOT 40,128
    -K: DRAW 15,0
130 PLOT 175,127+(K-1)/10:
    DRAW  INK 1;56,0: PLOT
    175,127+K/10: DRAW  INK
    1;56,0
135 PRINT  INK 0;AT 3,5;K;
    INK 0;AT 3,25;INT (K/10)
140 NEXT K
150 INK 0: PRINT AT 21,0;"[6
    *ESPACIO]OTRA VEZ?(S O
    N)[5*ESPACIO]"
160 IF INKEY$="S" THEN  RUN
170 IF INKEY$<>"N" THEN  GO
    TO 160
180 STOP
```

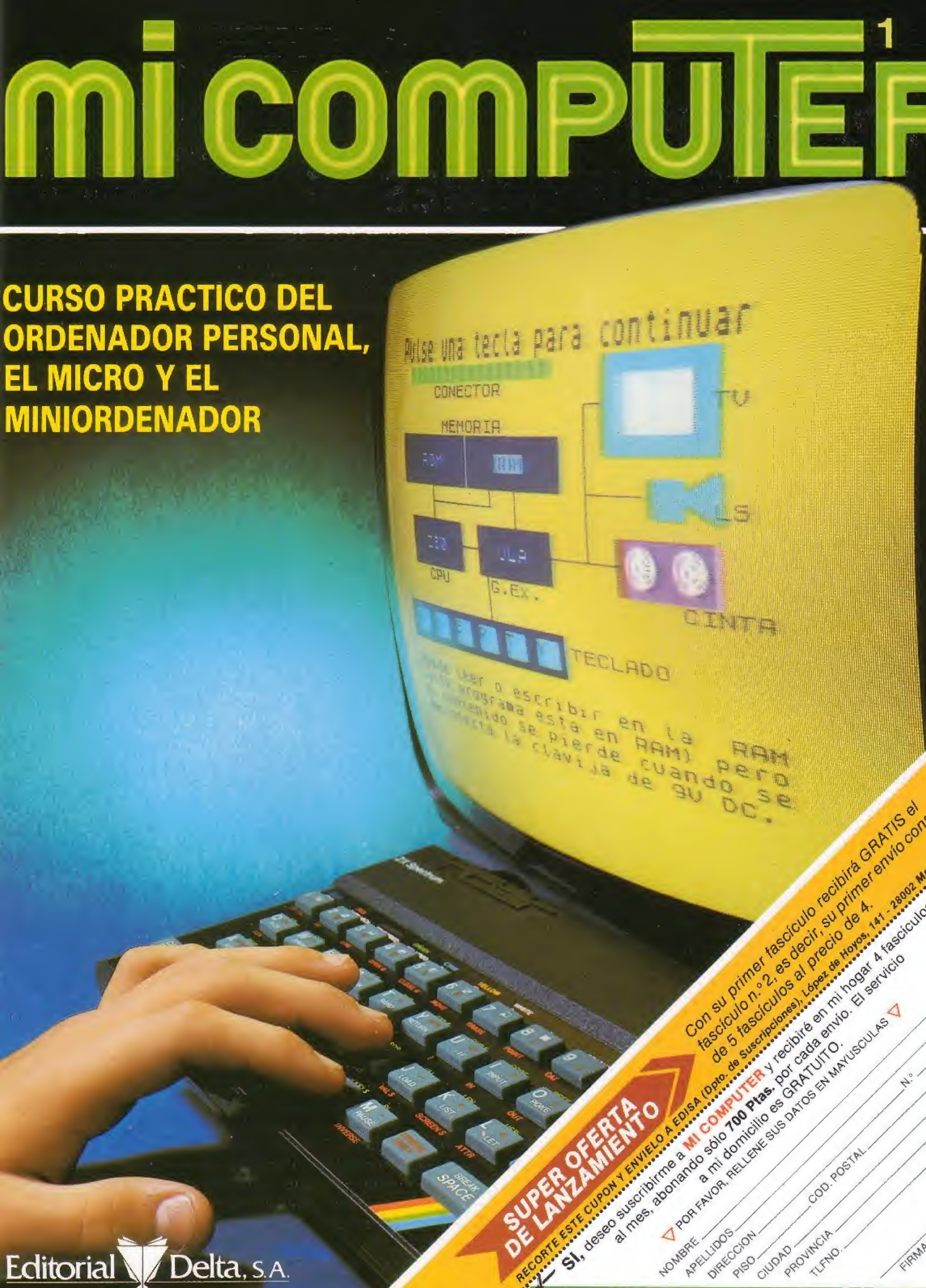
```
300 FOR N=6 TO 18: PRINT
    PAPER 1;AT N,5;"  ":
    NEXT N
310 FOR N=6 TO 18: PRINT
    PAPER 1;AT N,22;"[7*
    ESPACIO]": NEXT N
320 FOR N=18 TO 21: PRINT
    PAPER 1;AT N,5;"[23*
    ESPACIO]": NEXT N
330 PLOT 39,155: DRAW 0,-155
    : DRAW 192,0: DRAW 0,155
    : PLOT 56,155: DRAW 0,-
    124: DRAW 120,0: DRAW 0
    ,124
340 PLOT 40,127: DRAW -2,0:
    PRINT AT 6,3;"0": PLOT
    40,37: DRAW -2,0: PRINT
    AT 17,2;"90": PLOT 232,
    127: DRAW 2,0: PRINT AT
    6,30;"0"
350 PLOT 232,137: DRAW 2,0:
    PRINT AT 4,30;"9"
360 FOR N=127 TO 37 STEP -10
370 PLOT 40,N: DRAW -2,0:
    NEXT N
380 PLOT 232,132: DRAW 2,0
390 PLOT 120,35: DRAW 0,-35
400 PLOT 110,40: DRAW 20,0:
    DRAW -5,5: DRAW 5,-5:
    DRAW -5,-5
410 INK 7
440 RETURN
```

Al ejecutar este programa, en la línea 50 se produce un salto a una subrutina que dibuja el ariete. A continuación el programa te pide que especifiques el recorrido del pistón, que en un sistema de frenado sería equivalente al movimiento del pedal. Se produce entonces una animación que describe el movimiento del líquido en el ariete.

El desplazamiento es una medida del esfuerzo aplicado al pistón, pero la ventaja mecánica está determinada por los diámetros de los pistones motriz y de carga. Si los dos pistones tienen el mismo diámetro, no hay ventaja mecánica, pero a medida que se reduce el diámetro del pistón motriz aumenta ésta, de forma que es más fácil elevar la carga del lado derecho.

mi COMPUTER 1

CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR



Editorial Delta, S.A.

Código máquina

APROVECHA LAS INTERRUPCIONES

- EL ORDENADOR COMO RELOJ
- LA PRESENTACION
- UNA SENCILLA RUTINA EN C.M.
- ARRANQUE DEL RELOJ
- PUESTA EN HORA

Si dispones de un poco de tiempo, ensaya la siguiente rutina en código máquina, que te presentará el propio «reloj» interno de tu ordenador como si se tratase de un reloj digital siempre en hora

Tu ordenador dispone de un reloj interno que avanza con ritmo constante y sirve para regular sus operaciones. También tú puedes servirte de este reloj en un buen número de instrucciones del BASIC. Tal como ocurre con PAUSE.

Dichas instrucciones del BASIC hacen que el ordenador se ponga a contar hasta un número especificado, en centésimas o en cincuentaavos de segundo, dependiendo de la velocidad del reloj interno de la máquina. Hay otras muchas operaciones que utilizan el reloj de una forma análoga; así ocurre por ejemplo en los programas para ejecutar música, en los que se especifica la duración de cada nota.

TOMANDO TIEMPOS

De hecho, independientemente de que tu programa especifique o no de esta forma tan evidente la duración de las operaciones, el ordenador continuamente está consultando su reloj del que se sirve constantemente en la ejecución de los programas.

Es muy sencillo que el ordenador mida tiempos por tí. Para ello no tienes más que escribir un sencillo programa que escriba la hora, se pare durante un segundo, sume un segundo y vuelva a imprimir de nuevo. Si pruebas a hacerlo así, en seguida descubrirás que la pausa necesaria es realmente bastante más pequeña que un segundo, debido al tiempo que el ordenador necesita para realizar las operaciones de suma y PRINT.

Un reloj de este tipo tiene dos grandes inconvenientes. El primero de ellos es que solamente mantiene la hora mientras el ordenador está conectado. Esto puede que no sea un problema muy grave; si sólamente necesitas saber cuánto tiempo has estado trabajando en una determinada tarea, puede incluso ser una ventaja. Pero el segundo inconveniente es bastante más serio. Consiste en que en cuanto quieras utilizar el ordenador para otra cosa, el programa dejará de medir tiempos. Naturalmente, ello se debe a que no puedes ejecutar dos programas BASIC a la vez. La solución está en la utilización de una rutina en código máquina.

UN RELOJ DE INTERRUPCIONES

Al igual que otros programas en código máquina, que son ejecutados incluso cuando está corriendo un programa en BASIC, el reloj de código máquina que veremos a continuación hace uso de una rutina activada por interrupciones.

Durante todo el tiempo que la máquina está encendida la operación se ve constantemente interrumpida a intervalos regulares, durante una pequeña fracción de segundo. Esto ocurre incluso cuando se está ejecutando un programa en BASIC, ya que el ordenador tiene que examinar si se ha pulsado alguna tecla entre otras cosas. Por eso el programa BASIC se detiene mientras el ordenador explora el teclado y vuelve a ejecutarse nuevamente hasta que llega el momento de la siguiente interrupción.

Puedes incluir una rutina en código máquina anexa a esta exploración del teclado, de manera que se ejecute en las imperceptibles detenciones del programa en BASIC. El resultado es que hay dos programas que parecen estar ejecutándose al mismo tiempo.

Como la propia interrupción está controlada por el reloj del ordenador, resulta ideal para nuestros objetivos, ya que se puede hacer funcionar el reloj sin más que contar el número de interrupciones. La frecuencia de las interrupciones varía de unos ordenadores a otros, pero el principio utilizado es siempre el mismo. En el **Spectrum** las interrupciones se producen cada 1/50 de segundo.

El programa que viene a continuación configura un sencillo reloj digital que cuenta las horas, minutos y segundos a partir del momento en que el reloj es cargado y puesto en marcha. Puedes poner a cero la lectura de la pantalla, de forma que lo puedes utilizar como reloj de tiempo real o como cronógrafo.

El tiempo dado por este reloj no es absolutamente exacto. La rutina de inicialización de cada uno de los bucles requiere un cierto tiempo, pero se trata de millonésimas de segundo. Por eso cualquier error que se presente se deberá más bien a algún error del reloj interno de tu ordenador. Incluso así, el resultado no derivará más de un segundo por día. No obstante, si utilizas las instrucciones SAVE, LOAD o BEEP, el reloj se detendrá durante el tiempo que dure la operación correspondiente. La lectura digital se presenta constantemente en la esquina superior derecha de la pantalla, superponiéndose sobre cualquier otra cosa que hubiera allí. Si esto te causa problemas, puedes reconfigurar tu pantalla de modo que se elimine la primera línea.

La siguiente rutina vale tanto para máquinas de 16K como de 48K. Sin embargo no puedes utilizarla cuando esté conectada el **Interface 1**, ya que ello modificaría los vectores de interrupción.

Teclea para Spectrum

```
10 CLEAR 32319: LET TOTAL=0
20 FOR N=32320 TO 32554:
   READ A: POKE N,A: LET
   TOTAL=TOTAL+A: NEXT N
30 IF TOTAL<>24216 THEN
   PRINT "ERROR EN DATA":
   STOP
40 RANDOMIZE USR 32320
50 DATA 33,0,0,34,120,92,34,
   121,92,62,40,237,71,237,
   94,201,0,64,0,0
60 DATA 62,62,237,71,237,86,
   201,0,229,213,197,245,58,
   91,126,60,50,91,126,254
70 DATA 50,32,50,175,50,91,
   126,58,120,92,60,50,120,
   92,254,60,32,35,175,50
80 DATA 120,92,58,121,92,60,
   50,121,92,254,60,32,20,
   175,50,121,92,58,122,92
90 DATA 60,50,122,92,254,13,
   32,5,62,1,50,122,92,58,
   122,92,38,0,111,17
100 DATA 23,64,205,234,126,
   58,121,92,38,0,111,17,
   26,64,205,234,126,58,
   120,92
110 DATA 38,0,111,17,29,64,
   205,234,126,17,208,61,
   33,29,64,205,34,127,17,
   208
120 DATA 61,33,26,64,205,34,
   127,62,120,33,24,88,119,
   17,25,88,1,7,0,237
130 DATA 176,205,191,2,241,
   193,209,225,251,201,237,
   83,80,126,1,246,255,205,
   251,126
140 DATA 1,255,255,205,251,
   126,201,175,9,60,56,252,
   237,66,61,198,48,229,205
   ,21
150 DATA 127,33,80,126,52,42
   ,80,126,205,34,127,225,
   201,237,75,54,92,38,0,
   111
160 DATA 41,41,41,9,235,201,
   6,8,26,119,36,19,16,250,
   201
```

El código máquina contiene una serie de sentencias DATA que la línea 20 se encarga de POKEar en memoria. Como hay una gran cantidad de DATAs y es muy fácil cometer errores al copiar tantos números, en la línea 20 se ha previsto también una comprobación del total; si el resultado de la suma no es el correcto, la línea 30 detiene el programa y envía un mensaje de error invitándote a comprobar las cifras tecleadas.

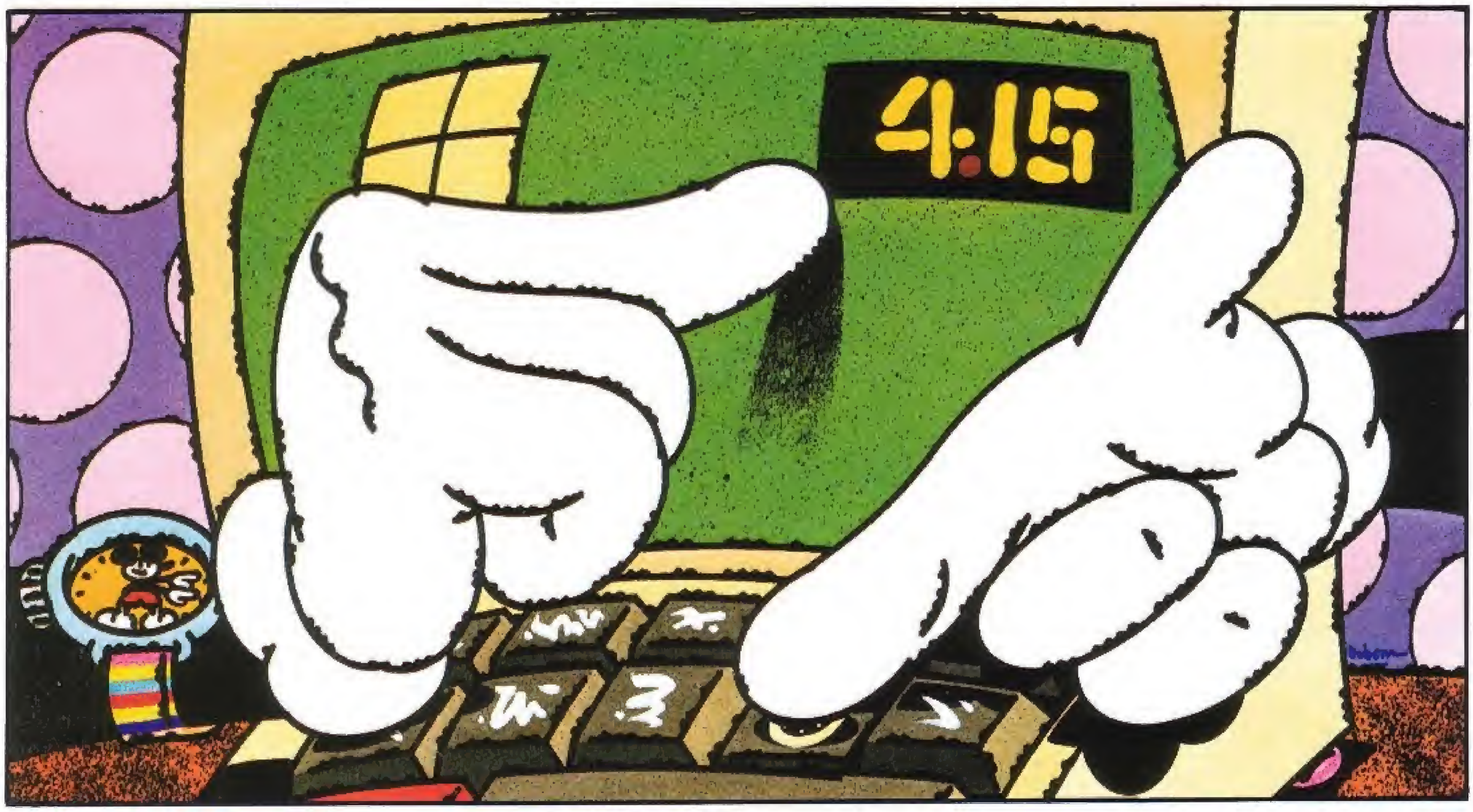
La línea 10 desplaza hacia abajo el área de comienzo del BASIC para proteger el código máquina, al que se llama automáticamente en la línea 40 al ejecutar este programa en BASIC. El reloj empieza en 00:00:00, pero puedes reinicializarlo con los siguientes POKEs:

```
POKE 23672,(segundos)
POKE 23673,(minutos)
POKE 23674,(horas)
```

Los números que siguen a los POKEs han de estar dentro de los márgenes permitidos —de 0 a 60 para minutos y segundos y de 1 a 12 para las horas—. Si quieres poner de nuevo a cero el reloj, es más rápido teclear lo siguiente:

```
RANDOMIZE USR 32320
```

con lo que se llama otra vez desde el principio a la rutina en código máquina. También necesitarás utilizar esta instrucción cuando ejecutes un NEW, con el que se inicializan todas las interrupciones.



Programación

ORDENADORES, PUZZLES Y MATEMATICAS

- TIPOS DE PUZZLES
- USO DEL ORDENADOR
- ECUACIONES SIMULTANEAS
- PRUEBA Y ERROR
- TRUCOS MATEMATICOS

Agudiza el ingenio y pon a prueba tu destreza en programación con estos difíciles rompecabezas. También aprenderás aquí unas cuantas técnicas nuevas para resolver sistemas de ecuaciones.

A estas alturas de nuestra andadura en INPUT, ya tienes una buena base en casi todas las técnicas de programación en BASIC. Dado un problema cualquiera, deberías ser capaz de escribir un programa para resolverlo (en el supuesto de que realmente pueda ser resuelto). Pero a menos que tengas un *hobby* especial o algún proyecto entre manos que haga uso del ordenador, con frecuencia es difícil pensar cosas con las que ponerse a hacer pruebas.

Los rompecabezas ofrecen un desafío que suele ser bienvenido, ya que constituyen una forma entretenida de practicar la programación y de mantener al día tus habilidades. Han sido muy populares durante millares de años. Se sabe que los egipcios disfrutaban resolviendo rompecabezas y que los antiguos griegos eran extremadamente aficionados a los rompecabezas y paradojas matemáticas y lógicas. De hecho muchos problemas que empezaron como un simple divertimento condujeron posteriormente a importantes descubrimientos en matemáticas y en otras ciencias.

La resolución de un problema con éxito te hará sentir una sensación de satisfacción y, si participas en alguno de los cada vez más abundantes concursos organizados por las revistas de informática, puedes ganar muchos premios. Pero la solución de estos rompecabezas te enseñará además sobre programación mucho más de lo que puedes aprender por simples lecturas sobre el tema ya que te obliga a trabajar con tus propios métodos y a poner a prueba tus propias ideas.

PUZZLES PARA ORDENADOR

Existen muchos tipos de rompecabezas, no todos de los cuales son adecuados para ser resueltos por medio de ordenador. Muchos de ellos requieren una idea feliz o un poco de razonamiento lógico. Un ejemplo clásico del tipo que requiere razonamiento lógico es el siguiente. Un hombre posee tres cosas: un lobo, una oveja y una berza, las cuales tiene que transportar a la orilla opuesta de un río. Pero el bote que utiliza es muy pequeño y sólo puede llevar una cosa cada vez. Y si deja en tierra el lobo y la oveja, el lobo matará la oveja, mientras que si deja a la oveja y la berza, la oveja se comerá la berza. ¿Cómo puede hacer para pasar a la otra orilla del río las tres cosas y tenerlas intactas?

Este enigma puede resolverse con un ordenador (si te gusta, puedes intentar escribir un programa que lo haga), pero es mucho más rápido hacerlo con papel y lápiz o directamente de memoria.

Los tipos de enigmas que mejor se resuelven por ordenador son los que contienen un cierto número de hechos y cifras, en los que se pide encontrar el valor de alguna incógnita. Otros que también funcionan bien sobre ordenador son los que exigen realizar tareas complicadas de tipo aritmético o geométrico. En tales casos es más rápido escribir un programa que resuelva el problema que intentar hacerlo por tí mismo.

Este artículo te enseñará la forma de resolver tres de los tipos más comunes de *puzzles*. Antes de mirar las soluciones, merece la pena que te entretengas en gastar algo de tiempo intentando resolverlos tú solo. Después pasa ya a consultar los programas y la explicación de su funcionamiento.

SIMPLIFICACION DEL PROGRAMA

Si no estás muy acostumbrado a resolver adivinanzas, al principio puede que la cosa te resulte bastante difícil. Se requiere un poco de práctica para separar la información esencial de la masa de palabras confusas, que sólo sirve para despistar. De hecho los enigmas del primer tipo se basan en las palabras que enmascaran lo que con frecuencia es un problema bastante sencillo. Cuando se le despoja de sus palabras, el enigma conduce habitualmente a la solución de un sistema de ecuaciones. Aquí tienes un sencillo problema que puedes resolver sin ordenador y que te ayudará a entender los principios que intervienen.

Un grupo de amigos entra en un bar y pide tres cafés y dos tés. La cuenta asciende a 176 pesetas. Al día siguiente se les agrega alguien más, y piden el doble de tés y un café menos. Esta vez la cuenta sube 16 pesetas más. ¿Cuánto cuesta la taza de té?

Si eliminas toda la información no esencial y utilizas letras en vez de palabras, el problema se reduce a $3c + 2t = 176$ y $2c + 4t = 192$.

A esto se le llama sistema de ecuaciones simultáneas, porque han de satisfacerse las dos al mismo tiempo. Para poder resolver un sistema de ecuaciones simultáneas se necesitan tantas ecuaciones como incógnitas. En este caso hay dos incógnitas a las que hemos llamado c y t, así como dos ecuaciones, por lo que el sistema en principio tiene solución. Además las ecuaciones son lineales ya que ninguna de las variables aparece elevada a una potencia mayor que 1. Por ejemplo $-3c^2 + 2t = 176$ es una ecuación no lineal que se resuelve por un método diferente.

Para resolver el problema puedes reordenar la segunda ecuación, lo que

te dará c = (192 − 4t)/2 y a continuación sustituir en la primera. Se obtiene 3(192 − 4t)/2 + 2t = 176. Reagrupando de nuevo los términos, se obtiene t = 28, por lo que una taza de té cuesta 28 pesetas.

Las ecuaciones con dos incógnitas como ésta son fáciles de resolver y las de tres incógnitas tampoco son demasiado difíciles. Pero para un mayor número de incógnitas el ordenador resulta decisivo a la hora de hacer por tí el trabajo más duro.

EL VENDEDOR DE SELLOS

Aquí tienes el primer enigma real que debes resolver.

Un vendedor de sellos tiene una caja con sellos extranjeros que piensa vender en paquetes de seis valores diferentes. El precio de los paquetes está codificado desde la A a la F. Seis jóvenes miembros de un club filatèlico se presentan a la vez dispuestos a gastarse todo el dinero que llevan en el bolsillo como se ve en el cuadro 1.

	A	B	C	D	E	F	COSTE
ELENA	6	2	3	1	1	2	341
JAIME	14	11	0	1	2	1	469
ANITA	0	1	3	6	4	3	598
MAR	5	3	5	2	1	1	376
LUIS	12	1	4	4	3	2	587
CLARA	8	0	1	1	0	3	293

La pregunta es: ¿Cuál era el precio de cada paquete?

El intentar resolver este sistema de ecuaciones eliminando una variable en cada paso, es posible, pero requeriría mucho tiempo y se presta a que se cometan errores de cálculo. Con el siguiente programa podrás resolver esta adivinanza así como cualquier otro sistema de ecuaciones lineales simultáneas, las cuales se presentan en muchos tipos de problemas y no sólo en la resolución de acertijos.

Existen varias formas de resolver sistemas de ecuaciones como éste, y puedes encontrar métodos diferentes, pero el programa que sigue es rápido y relativamente corto:

Teclea

```
10 INPUT "NUMERO DE FILAS? "
   ;R
15 LET C=R+1
20 DIM A(R,C): DIM B(R,
   C): DIM A$(C-1,20)
30 FOR K=1 TO C-1
40 INPUT "NOMBRES DE LAS
   COLUMNAS? ",A$(K)
50 NEXT K
60 FOR J=1 TO R
70 PRINT : PRINT "VALORES
   DE LAS FILA? ";J
```

```
80 FOR K=1 TO C
90 INPUT A(J,K)
95 PRINT A(J,K)
100 LET B(J,K)=A(J,K)
110 NEXT K: NEXT J
120 FOR L=1 TO R
130 GO SUB 230
140 GO SUB 280
150 NEXT L
160 CLS
170 FOR K=1 TO C-1: PRINT
    AT 1,4*K-4;A$(K): NEXT K
180 FOR J=1 TO R: FOR K=1 TO
    C: PRINT AT 1+J,K*4-4;B
    (J,K)
190 NEXT K: NEXT J
200 PRINT "RESPUESTAS:-"
210 FOR K=1 TO C-1: PRINT AT
    4+R,K*4-4;A(K,C): NEXT K
220 STOP
240 LET D=A(L,L)
250 FOR K=1 TO C
260 LET A(L,K)=A(L,K)/D
270 NEXT K: RETURN
290 FOR J=1 TO R
300 IF J=L THEN NEXT J:
    RETURN
310 LET F=A(J,L)
320 FOR K=1 TO C
330 LET A(J,K)=A(J,K)-F*A(L,
    K)
340 NEXT K: NEXT J: RETURN
```

La primera parte del programa, entre las líneas 10 y 110, te permite introducir los datos, mientras que la segunda parte, situada entre las líneas 120 y 150 llama a dos subrutinas (o procedimientos) para calcular la respuesta, la cual se imprime entonces en las líneas 160 a 220.

Los valores se cargan en la matriz A(J,K) a razón de una fila cada vez. En este programa, J se refiere a las filas y K a las columnas, lo cual has de tener siempre en la mente al leer el resto del programa. La matriz A(J,K) se copia sobre una matriz idéntica B(J,K) de forma que se pueden imprimir los valores originales junto con la respuesta en la línea 180.

El cálculo avanza de fila en fila, controlado por el bucle que se extiende entre las líneas 120 y 150. Antes de hacer otras cosas, la rutina de las líneas 230 a 270 extrae el elemento diagonal de la fila (es decir A(1,1), A(2,2), etc.) y divide cada elemento de dicha fila por ese valor. El valor del elemento diagonal resultante es ahora igual a uno, es decir, ha quedado normalizado.

La siguiente rutina es la encargada de hacer la mayor parte del trabajo de este programa. Aunque sólo contiene seis líneas, resulta muy difícil examinar lo que hace. La mejor manera de entenderlo es examinar a fondo un ejemplo real (aunque corto) siguiendo uno por uno los pasos que va dando el ordenador. En esencia funciona de la siguiente manera. Supongamos que ya has normalizado la fila 1, por lo que L es 1. La línea 290 toma entonces cada fila diferente de la 1 (línea 300), por lo que en este caso empieza por la 2. La línea 310 toma el primer elemento de la fila 2 y le llama F. Todavía en la fila 2, la línea 320 va recorriendo todas sus columnas, y la línea 330 multiplica F por el elemento correspondiente de cada columna, pero de la fila 1, restándolo del elemento de la misma columna en la fila 2. Esto mismo lo va haciendo con las filas 3, 4, 5 y 6. Seguidamente empieza otra vez con L igual a 2.

Al final de este proceso, los elementos diagonales de cada fila siguen siendo y todos los demás elementos son ceros. Por eso se pueden leer con facilidad los valores en la columna de la derecha.

LOS REGALOS DE NAVIDAD

El programa que acabamos de ver sirve para resolver cualquier sistema de ecuaciones lineales simultáneas en el que hay tantas ecuaciones como incógnitas. Pero existen rompecabezas en los que las cosas se organizan de tal manera que no hay ecuaciones suficientes, por lo que no se pueden resolver de esta forma. De hecho no existe una solución única y el truco consiste en seleccionar la respuesta más plausible entre todas las soluciones disponibles. Normalmente el enigma contendrá una clave que te ayudará. También podría ocurrir que las ecuaciones resultaran ser no lineales.

Ensaya ahora el siguiente enigma. En Navidad, el tío Alberto, que es un matemático un poco excéntrico, explicó a sus dos jóvenes sobrinos que le daría a cada uno tantos paquetes como la edad que tenían (contando sólo los años). También les dijo que cada paquete contenía el mismo número de sobres que la edad de cada uno, y que cada sobre contenía un número de pesetas igual a la edad de cada niño. Después de preparar los regalos, se le oyó murmurar entre dientes que el siguiente año la cosa le iba a costar 500 pesetas más. ¿Qué edades tenían los dos niños?

Suprimiendo las palabras y llamando A y B a las edades de los niños, así como M a la cantidad de dinero gastada en este año, la solución del enigma queda reducida a la de las ecuaciones siguientes:

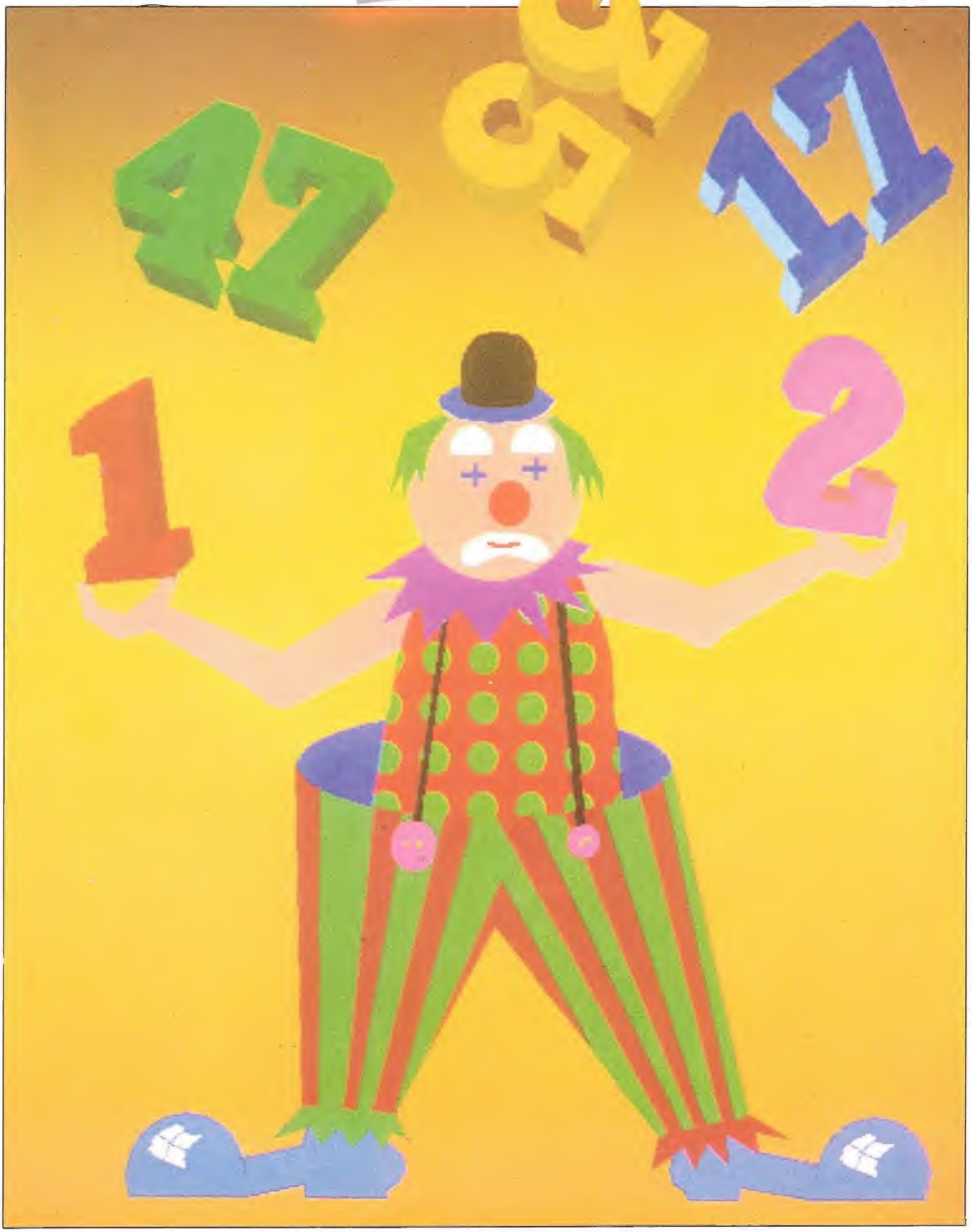
$A \uparrow 3 + B \uparrow 3 = M;$

$(A + 1) \uparrow 3 + (B + 1) \uparrow 3 = M + 500$

Tenemos tres incógnitas: A, B y M, pero sólo dos ecuaciones. Sin tener un ordenador, podrías utilizar un método de prueba y error para ir ensayando diferentes valores de A y B, y observando si las ecuaciones se satisfacen. El ordenador trabaja de una forma parecida, salvo que naturalmente puede hacer varios cientos o miles de ensayos de valores diferentes en un tiempo muy corto y sin cometer errores.

El primer paso en la resolución del problema es buscar en el enunciado del enigma alguna pista suplementaria que te pueda ayudar a limitar el margen de valores que debes ensayar. Como se habla de «jóvenes» sobrinos, no es probable que tengan más de 14 años, y es muy probable que al menos tengan tres años. El programa para resolver el problema es muy corto:

```
10 FOR A=3 TO 14
20 FOR B=A TO 14
30 LET M=A^3+B^3
40 LET N=(A+1)^3+(B+1)^3
50 IF ABS (M+500-N)<.01 THEN
   PRINT "A=";A,"B=";B
60 NEXT B
70 NEXT A
```


47
57
17
1
2

ECUACIONES SINGULARES

Hay algunas situaciones especiales, con frecuencia bastante tontas, en que los sistemas de ecuaciones simultáneas tienen o una solución ambigua o carecen por completo de solución. Las ecuaciones x = 2 y x = 3 obviamente no pueden satisfacerse al mismo tiempo, e incluso cuando tiene tantas ecuaciones como incógnitas, como en el caso x + y = 3; x + y = 1 no pueden cumplirse simultáneamente.

Otro caso son las ecuaciones x + y = 2; 2x + 2y = 4. En este caso se trata de dos ecuaciones que no son independientes; esencialmente se trata de la misma ecuación, existiendo muchas soluciones posibles.

Cuando cuentes con el número de ecuaciones adecuado y siga habiendo incompatibilidades o las ecuaciones no sean independientes, se dice que el sistema es singular.

El caso de las ecuaciones dependientes se puede resolver utilizando un método de prueba y error, como hacíamos en el caso de los regalos de Navidad. Pero al ser posibles varias soluciones necesitas alguna información adicional para seleccionar la correcta.

Al ejecutar este programa obtendrás solamente una respuesta, debido a las restricciones introducidas en las líneas 10 y 20. Si en el enigma no se hubiera especificado que los sobrinos son jóvenes, el bucle FOR ... NEXT tendría que extenderse más. Digamos de paso que la condición impuesta en la línea 50 evita problemas con los errores de redondeo; lo que realmente se está comprobando es si se cumple que M + 50 es igual a N.

Este método debe funcionar para todos los problemas con más incógnitas que ecuaciones. En otros tipos de problemas, puede que te sea necesario hacer hipótesis acerca de los valores de más variables, o tal vez incluir más condiciones IF ... THEN del tipo de la que aparece en la línea 50. Con un número muy elevado de incógnitas o de condiciones el programa puede tardar muchos minutos o incluso horas en ejecutarse, aunque seguirá siendo resoluble.

MAGIA MATEMATICA

El tercer tipo de rompecabezas que aparecen en las revistas de informática son los problemas de números. Suelen plantear cuestiones de aritmética aparentemente sencillas que resultan extraordinariamente laboriosas de resolver a menos que se utilice un ordenador.

Aquí tienes un par de ejemplos.

Existe un número de cuatro cifras que cuando se invierte y se multiplica por un entero se convierte en el número original: 8712 = 4*2178. El problema es encontrar si existen más números con esta misma propiedad, y en caso afirmativo calcularlos.

Otro número interesante es el 987654321. Este número es un múltiplo exacto de 17 y utiliza todos los dígitos del 1 al 9. Trata de encontrar el número inferior más proximo que tenga estas mismas propiedades.

Las soluciones a este tipo de problemas se basan en considerar el número como una colección de cifras más que como un valor numérico. Puedes, dividir el número en unidades, decenas, centenas, etc, o tratarlo como una cadena de cifras y separar éstas con ayuda de los comandos de tratamiento de cadenas.

El primer problema se resuelve mediante el primer método. Si representamos el número por ABCD, da origen a la siguiente ecuación:

1000*A + 100*B + 10*C + D = X*(1000*D + 100*C + 10*B + A)

cuyas incógnitas tienes que calcular. Como de costumbre la principal dificultad consiste en decidir qué valores se ensayan. A puede variar entre 1 y 9, no pudiendo ser cero ya que en ese caso tendrías un número de tres cifras. B y C pueden variar entre 0 y 9. El factor de multiplicación X debe ser al menos 2 y no debe superar a 10/D, para que el segundo miembro de la ecuación sea un número de cinco cifras. Por la misma razón D no puede ser mayor que 4. Ahora puedes ya escribir el programa para resolver el problema:

```
10 FOR A=1 TO 9
20 FOR B=0 TO 9
30 FOR C=0 TO 9
40 FOR D=1 TO 4
50 FOR X=2 TO INT (9.9/D)
60 LET J=1000*A+100*B+10*C+D
70 LET K=1000*D+100*C+10*B+A
80 IF X*K=J THEN  PRINT J;
   "=";X;"*";K
90 NEXT X: NEXT D: NEXT C:
   NEXT B: NEXT A
```

Teclea el programa, a continuación teclea RUN y siéntate a esperar porque llevará bastante tiempo la comprobación de todas las combinaciones posibles.

La solución del segundo problema trata a los números como cadenas.

```
10 LET M=98765432
20 LET M=M-27
30 LET M$=STR$ M
40 LET F=0
50 FOR P=2 TO 9
60 LET P$=STR$ P
70 LET X=0: FOR K=1 TO 8:
   IF P$=M$(K TO K) THEN
   LET X=K
75 NEXT K
80 IF X=0 THEN  LET F=F+1
90 NEXT P: IF F=0 THEN
   PRINT M$: STOP
100 GO TO 20
```

El programa parte de 987654321 y va contando hacia abajo de 17 en 17, convirtiendo cada múltiplo de 17 en una cadena de números a la que llama M$. Las líneas 50 y 60 convierten cada dígito del 1 al 9 en una cadena y a continuación la línea 70 comprueba si dicha cadena figura en M$. Si hay algún dígito que no esté en M$, hay un indicador F que se incrementa una unidad. Sólo se imprime M$ cuando contiene todos los dígitos del 1 al 9.

SIMULADOR DE VUELO

- LA SIMULACION DE VUELO
- PROGRAMAS DE ENTRENAMIENTO
- VUELO MEDIANTE INSTRUMENTOS
- EL PANEL DE INSTRUMENTOS
- PERDIDA DE VELOCIDAD

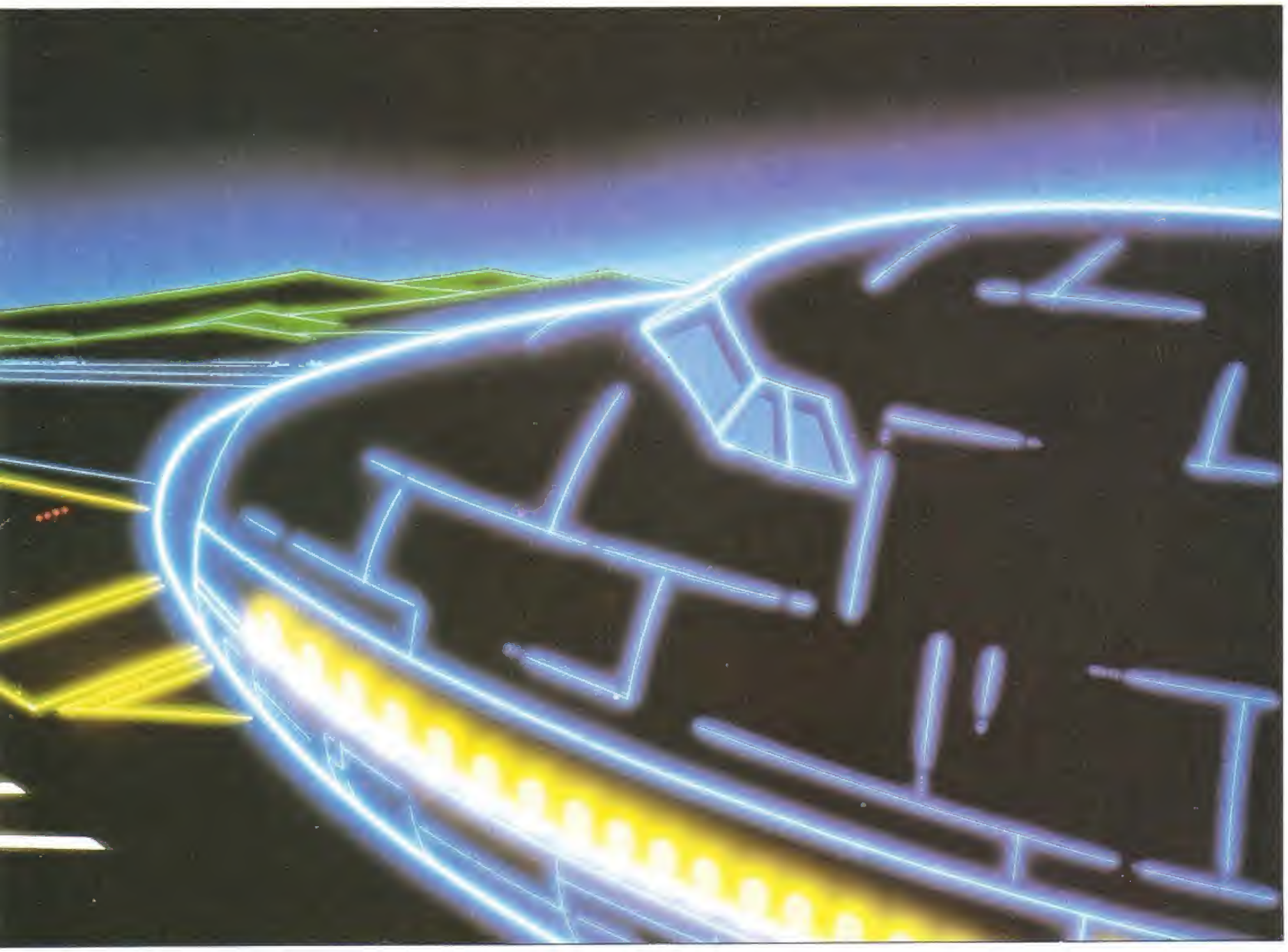
Este programa de simulación de vuelo es similar a los que se emplean en las escuelas de vuelo para enseñar a los pilotos cómo tienen que volar utilizando únicamente sus instrumentos; en la primera parte se reproduce la cabina

Los programas de juegos varían desde una fantasía desbordante que supone la entrada a mundos imaginarios y la participación en aventuras, hasta la simulación de situaciones de la vida real. Esto te permite poner a prueba tu capacidad en situaciones potencialmente peligrosas, sin hacerte daño o perder totalmente millones de pesetas en costosos equipos.

Los programas de simulación de vuelo contienen un elemento de fantasía: tú solo en la cabina, con toda la tripulación aquejada de una misteriosa enfermedad, con una sola mano consigues hacer que el avión tome tierra felizmente. Pero los sofisticados programas de este tipo tienen un uso práctico real, hasta el punto de que casi todas las principales compañías de líneas aéreas y escuelas de vuelo los utilizan con regularidad.

SIMULADORES DE ENTRENAMIENTO

En el extremo superior de la escala está la simulación total, la llamada «Fase 3» en la terminología de las administraciones de aviación civil, que te permite experimentar las mismas sensaciones que un piloto en un avión de verdad. Tú ves lo mismo que él ve a



través de la ventanilla de la carlinga (incluyendo una pequeña diferencia de ángulo en el punto de vista para la posición del copiloto); sentirás lo mismo que él siente en los despegues y en los aterrizajes, así como las turbulencias; oirás lo mismo que él oye, incluyendo las indicaciones del control de tráfico aéreo. En teoría un piloto puede completar todo su entrenamiento en uno de estos simuladores y obtener su licencia sin tener que abandonar el suelo para nada.

SIMULADORES DE SOBREMESA

En el extremo opuesto de la escala están los programas de simulación de vuelo muy parecidos al que veremos a continuación.

Las unidades de sobremesa se pueden «volar» en el interior de un aula, y resultan útiles para la enseñanza de los procedimientos de cabina y para desarrollar la rapidez de reflejos de los pilotos.

Resultan además esenciales para la enseñanza del vuelo por instrumentos, una técnica que permite al piloto navegar apoyándose únicamente en el panel de instrumentos, algo que todo piloto ha de hacer cuando las condiciones meteorológicas son malas.

LO QUE HACE EL PROGRAMA

Este artículo consta de tres partes y en él se presenta un programa de simulación de vuelo en el que se supone que te has hecho cargo del control del avión cuando se encuentra a 2000 metros de altura y a 20000 metros de distancia de la pista de aterrizaje que constituye el objetivo. Por la ventanilla de la cabina es muy poco lo que puedes ver, solamente el horizonte, cuando hay visibilidad, y un punto distante que es la pista de aterrizaje, por lo cual como piloto sensato que eres, tendrás que confiar en tu experiencia y atendiendo a lo que te indique el panel de instrumentos ponerte a salvo en tierra a tí y a tus pasajeros.

LOS INSTRUMENTOS

En tu panel de instrumentos hay cuatro *diales*. El primero te informa de la velocidad del aire. Este valor cambia según que estés picando (tu velocidad aumenta), elevándote (tu velocidad disminuye) o modificando la potencia de los motores. Un contador que hay debajo del dial de la velocidad del aire te indica la marcación de tu brújula.

El segundo *dial* te muestra dónde está el horizonte en relación con tu aeroplano. Esto significa que aunque el horizonte no resulte visible a través de la ventanilla de la carlinga, todavía sigues sabiendo dónde está. El contador que hay debajo de este *dial* te indica la marcación de la pista de aterrizaje.

El tercer *dial* proporciona una lectura de la altitud. Tiene dos manecillas, una para los miles y otra para los cientos. El contador que hay debajo calcula la deriva del avión; como la pista de aterrizaje tiene una anchura de 100 metros una deriva de +50 o −50 hará que la pierdas.

El último *dial* indica la velocidad del motor en revoluciones por minuto. El contador que hay debajo permite conocer la distancia a la que te encuentras del centro de la pista.

ATERRIZAJE

Aunque hay programas de simuladores de vuelo en los que la imagen que se ve a través de la ventanilla de la cabina se va haciendo más nítida a medida que progresa la aproximación, no es éste nuestro caso. Por ello debes centrar la imagen radar de la pista de aterrizaje.

En el momento en que asumes el control, las condiciones meteorológicas son buenas y la pista se encuentra en la dirección norte. El aterrizaje en estas condiciones no es difícil y el juego perdería en seguida su atractivo si no se pudieran modificar estas condiciones. Para añadir más dificultad, puedes especificar la velocidad y la dirección del viento: por ejemplo un furioso vendaval que sople de lado te pondrá las cosas mucho más difíciles.

MOVIENDO EL AVION

El margen de los controles de que

dispones se aproxima bastante a los controles de un avión de verdad, aunque estés pulsando teclas en vez de utilizar un *joystick*.

En una aeronave real, el control de la elevación —los movimientos hacia arriba y hacia abajo— se hace moviendo el *joystick* hacia atrás o adelante, lo que hace que se muevan los elevadores del plano de cola hacia arriba o abajo. Tú vas a utilizar dos teclas para conseguir este mismo efecto; en la tercera sección de este artículo teclearás la parte de programa que se ocupa de esto.

El balance del aeroplano —el movimiento lateral— se controla moviendo el *joystick* de un lado a otro. Esto

hace que se muevan los alerones, que son las superficies de control que hay en las alas. También en este caso utilizarás dos teclas para hacer que se muevan hacia la derecha o hacia la izquierda.

Tus dos últimos controles te permiten acelerar o retardar la velocidad del motor, lo cual resulta esencial para controlar los tiempos en la maniobra de aterrizaje y para asegurarte de que no entrarás en pérdida de velocidad.

PERDIDA DE VELOCIDAD

Cuando la velocidad de un avión cae por debajo de un determinado valor, se dice que entra en pérdida de velocidad y en ese momento el avión empieza a caer como si fuera una piedra. En este programa, cuando la velocidad de tu avión cae por debajo de los 30 metros por segundo, el aeroplano caerá en picado abruptamente. Si tenías bastante altura en el momento en que esto ocurre, una acción rápida te puede salvar, pero una entrada en pérdida es algo que aterroriza a cualquier piloto.

DIVISION DEL PROGRAMA

El programa es demasiado largo y complejo como para darlo todo de una sola vez, por lo que lo hemos dividido en tres partes.

Lo que se hace en esta primera parte es configurar la pantalla para mostrar el interior de la cabina, con su ventanilla, los cuatro *diales* debidamente etiquetados y los letreros de los contadores.

Los comandos que intervienen serán familiares para la mayoría de vosotros por haberlos visto ya en otros programas.

La parte de programa introducida en la parte dos, hace posible que los *diales* y contadores sean sensibles al movimiento del avión y hay un comando temporal que hace que éste vuele aleatoriamente sin que haya un piloto que actúe sobre los controles, para que puedas ver funcionando el panel de instrumentos. La sección final te permite tomar el control del avión y realizar una estimación de tu técnica de aterrizaje para que puedas juzgar tus progresos.

DIBUJANDO LA CABINA

Para dibujar la cabina, teclea en tu ordenador la siguiente parte de programa.

Teclea

```
1 POKE 23658,8
110 GO TO 5000
5000 LET PP=-1: LET
     RR=-1
5010 LET C=PI/180: LET PY=-
     20000: LET PZ=2000:
     LET AS=150
5110 PLOT 10,175: DRAW 235,
     0: DRAW 0,-90: DRAW
     -235,0: DRAW
     0,90
5120 FOR K=0 TO 3: CIRCLE
     35+K*60,50,20: NEXT K
5130 PRINT AT 12,2;"VELOC[3
     *ESPACIO]HORZN[3*
     ESPACIO]ALT[4*ESPACIO]
     RPM"
5150 PRINT AT 20,0;"SITUACI
     PISTA  DERIV
     DISTANCI"
5170 PLOT 87,50: DRAW 5,0:
     DRAW 3,-3: DRAW 3,3:
     DRAW 5,0
5180 LET X=35: LET Y=50: GO
     SUB 7000: LET X=155:
     GO SUB 7000: LET X=215:
     GO SUB 7000
6900 STOP
7000 FOR K=0 TO 2*PI STEP
     PI/5: PLOT X+17*SIN K,
     Y+17*COS K: DRAW 2*SIN
     K,2*COS K: NEXT K:
     RETURN
```

El POKE de la línea 1 pone el ordenador en modo de mayúsculas. Las líneas 5000 y 5010 sitúan el aeroplano en su posición en vuelo: 2000 metros de altura, 20000 metros hasta su punto de destino y sin movimiento; la próxima vez teclearás las líneas que hacen que se mueva.

La línea 5110 dibuja la ventanilla de la cabina, mientras que los *diales* se dibujan en la línea 5120 utilizando un bucle FOR ... NEXT. Los letreros de los *diales* y los contadores se generan en las líneas 5130 y 5150. La línea 5170 dibuja el diagrama de un avión en el *dial* del horizonte; el horizonte artificial no se dibujará hasta el próximo número de nuestro coleccionable. La línea 5180 y el GOSUB a la subrutina que comienza en la 7000 definen los centros de los *diales* y dibujan los indicadores necesarios: Velocidad del aire, Altitud y RPM; para ello se utilizan las funciones SIN y COS.

Si ejecutas ahora el programa, te aparecerá en la pantalla tu cabina simulada de avión.

DESPEGA PARA TU PRIMER VUELO

- VUELO CON EL PILOTO AUTOMATICO
- APROXIMACION A LA PISTA
- DE ATERRIZAJE
- DIBUJANDO LA TRAYECTORIA
- EL PANEL DE INSTRUMENTOS

En la segunda parte del simulador de vuelo, puedes arrancar los motores y obsevar cómo adquiere vida tu panel de instrumentos. Pero ten cuidado: ¡El piloto automático se ha vuelto loco!

En la primera parte de este artículo has tecleado las líneas que reproducían sobre la pantalla el interior de la cabina de vuelo.

En esta parte verás adquirir movimiento a tu avión y cómo adquiere vida el panel de instrumentos; de modo que, aunque todavía no has tomado los mandos, puedes ver cómo responde el panel de instrumentos del avión ante los movimientos del mismo.

EL AVION VUELA

Esta es la parte más larga del programa, con bastante diferencia. Hay una larga serie de variables interdependientes que tienen que ser constantemente actualizadas para controlar el avance del avión. Además hay que volver a dibujar constantemente el panel de instrumentos a medida que van cambiando la posición y la altura del avión, ya que los *diales* tienen que ir describiendo el movimiento.

LA APROXIMACION A LA PISTA

Hay una imagen radar de la pista de aterrizaje que te muestra el ángulo con el que te estás aproximando.

DIBUJO DE LA TRAYECTORIA

Para poder dibujar con precisión la posición del avión, hay que tener en cuenta muchos factores. Por ejemplo, la dirección en la que estás volando, se ve afectada por la dirección del viento y por el balance del avión. La velocidad de avance depende parcialmente de la velocidad del viento. La altura que vas perdiendo, o ganando, está relacionada con la velocidad de vuelo, etc.

Para poder actualizar los *diales* y los contadores, hay que hacer una estimación de las variables que van cambiando, con arreglo a la forma en que afectan a las lecturas, después de lo cual se puede hacer de nuevo el dibujo.

Teclea

```
2 LET WY=0: LET WX=0: LET GZ
  =0: LET GY=0: LET GX=0
5 LET RW=0: LET Y1=120: LET
  Y2=120: LET Y3=40: LET Y4=
  40: LET POW=0: LET GC=0
  : LET RB=0: LET LL=0: LET
  YC=0: LET AD=0: LET ST=0:
  LET RL=0: LET BC=0: LET
  NC=0: LET PT=0: LET PX=0:
  LET VZ=0: LET VY=0: LET
  VX=0
500 LET RA=AD*C: LET VX=AS*
    SIN RA
510 LET VY=AS*COS RA: RETURN
1000 LET PZ=PZ+GZ: LET PY=PY
     +GY: LET PX=PX+GX
```

```
1025 IF ST=1 THEN  PRINT
     OVER 1;AT 4,12;
     "EN PERDIDA": LET ST=0:
     GO TO 1040
1030 IF AS<30 THEN GO SUB
     1500
1040 LET AD=AD+RL: IF AD<0
     THEN  LET AD=AD+360
1050 IF AD>359 THEN  LET AD
     =AD-360
1060 LET VZ=AS*SIN (PT*C)-
     10+AS/15
1070 LET GZ=VZ: LET GY=VY+WY
     : LET GX=VX+WX
1080 IF VY=0 THEN  LET GD=-
     PI/2: GO TO 1100
1090 LET GD=-ATN (VX/VY)/C
1100 GO SUB 500
1110 RETURN
1500 LET ST=1: PRINT  OVER 1
     ;AT 4,12;"EN PERDIDA":
     FOR M=1 TO 4: FOR N=20
     TO -20 STEP -4: BEEP
     .01,N: NEXT N: NEXT M
1510 LET RL=INT (RND*21)-9:
     LET PT=-21-INT (RND*5)
1520 RETURN
2180 IF GC<>0 THEN  GO SUB
     2200
2190 LET AS=AS+16*(TC*30-AS
     -8*PT)/AS: GO SUB 2200:
     GO TO 2205
2200 PLOT 35,50: DRAW  OVER
     1;15*SIN (AS*PI/200),15
     *COS (AS*PI/200):
     RETURN
2205 IF GC<>0 THEN  PLOT 155
     ,50: DRAW  OVER 1;10*
     SIN (TN*PI/5),10*COS
     (TN*PI/5): PLOT 155,50:
     DRAW  OVER 1;15*SIN (UN
     *PI/500),15*COS (UN*PI/
     500)
2210 LET TN=PZ/1000: LET UN=
     PZ-1000*INT TN: PLOT
     155,50: DRAW  OVER 1;10
     *SIN (TN*PI/5),10*COS
     (TN*PI/5): PLOT 155,50:
     DRAW  OVER 1;15*SIN (UN
     *PI/500),15*COS (UN*PI/
     500)
2220 IF GC<>0 THEN  GO SUB
     2230
2225 IF POW=-1 AND TC>.2
     THEN  LET TC=TC-.2
2226 IF POW=1 AND TC<8.8
     THEN  LET TC=TC+.2
2228 GO SUB 2230: GO TO 2240
2230 PLOT 215,50: DRAW  OVER
     1;15*SIN (TC*PI/5),15*
     COS (TC*PI/5): RETURN
2240 PRINT AT 21,2;ABS INT
     AD;"  "
2250 IF PY=0 THEN  LET RB=0:
     GO TO 2260
2255 LET RB=ATN (PX/PY)/C:
     IF PY>0 THEN  LET RB=
     RB+180
2260 IF RB<0 THEN  LET RB=
     RB+360
2270 PRINT AT 21,10;INT RB;
     "  ";AT 21,18;ABS INT
     PX;"  "
2280 PRINT AT 21,25;INT
     (SQR (PY*PY+PX*PX));
     " "
2290 IF (Y1<=110 AND Y2<=
     110) OR (Y1>=130 AND
     Y2>=130) THEN  GO TO
     2300
2295 IF GC<>0 THEN  PLOT
     OVER 1;X1,168-Y1: DRAW
     OVER 1;X2-PEEK 23677,
     168-Y2-PEEK 23678
2300 LET YC=120+(PT/3):
     LET X1=80: LET X2=110:
     LET Y1=YC+17*TAN (RL*2
     *C): LET Y2=YC-17*TAN
     (RL*2*C)
2310 IF (YC<110 OR YC>130)
     AND RL=0 THEN  GO TO
     2376
2320 IF Y1<110 THEN  LET X1
     =95-(95-X1)*(110-YC)/
     (Y1-YC): LET Y1=110:
     GO TO 2340
2330 IF Y1>130 THEN LET X1
     =95-(95-X1)*(130-YC)/
     (Y1-YC): LET Y1=130
2340 IF Y2<110 THEN  LET X2
     =95-(95-X2)*(110-YC)/
     (Y2-YC): LET Y2=110:
     GO TO 2360
2350 IF Y2>130 THEN  LET X2
     =95-(95-X2)*(130-YC)/(
     Y2-YC): LET Y2=130
2360 IF X1<80 OR X2>110 THEN
     GO TO 2376
2370 PLOT  OVER 1;X1,168-Y1:
     DRAW  OVER 1;X2-PEEK
     23677,168-Y2-PEEK 23678
2376 IF (RL=RR AND PP=PT)
     THEN  GO TO 2500
2377 IF (Y3<=2 AND Y4<=2) OR
     (Y3>=90 AND Y4>=90)
     THEN  GO TO 2380
2378 IF GC<>0 THEN  PLOT
     OVER 1;X3,176-Y3: DRAW
     OVER 1;X4-PEEK 23677,
```

```
     (176-Y4)-PEEK 23678
2380 LET YC=33+PT*4: LET X3
     =11: LET X4=244: LET
     Y3=YC+118*TAN (RL*2*C)
     : LET Y4=YC-118*TAN
     (RL*2*C)
2390 IF (YC<2 OR YC>90) AND
```

```
    RL=0 THEN  GO TO 2450
2400 IF Y3<2 THEN  LET X3=
    128-(128-X3)*(2-YC)/(
    Y3-YC): LET Y3=2: GO
    TO 2420
2410 IF Y3>90 THEN  LET X3=
    128-(128-X3)*(90-YC)/
    (Y3-YC): LET Y3=90
2420 IF Y4<2 THEN  LET X4=
    128-(128-X4)*(2-YC)/(Y4
    -YC): LET Y4=2: GO TO
    2440
2430 IF Y4>90 THEN  LET X4=
    128-(128-X4)*(90-YC)/(
    Y4-YC): LET Y4=90
2440 IF X3<11 OR X4>244 THEN
    GO TO 2500
2445 OVER 1: PLOT X3,176-Y3:
    DRAW X4-PEEK 23677,(176
    -Y4)-PEEK 23678: OVER 0
2500 GO SUB 8000
2505 IF GC=0 THEN  LET GC=1
2510 LET RR=RL: LET PP=PT:
    RETURN
5080 LET GZ=VZ: LET GY=VY+WY
    : LET GX=VX+WX
5090 LET TC=5
5100 LET RT=3: LET TP=5: LET
    WR=50
5500 IF INT (RND*5)=1 THEN
    LET RL=RL+INT (RND*5)-2
    : IF INT (RND*5)=1 THEN
    LET PT=PT+3-INT (RND*2)
    +1*2
5510 GO SUB 1000: IF PZ<0
    THEN  GO TO 5530
5520 GO SUB 2180: GO TO
    5500
5530 GO TO 5500
8000 IF GC<>0 THEN PLOT 127,
    174:DRAW OVER
    1;OX,OY
8010 LET OX=16*SIN (RB*(PI/
    180)): LET OY=-(16*ABS
    COS (RB*(PI/180)))
8020 PLOT 127,174: DRAW
    OVER 1;OX,OY
8025 LET WB=AD:
    IF AD>180
    THEN  LET WB=WB-360
8026 IF RB>180 THEN  LET WB
    =WB+360-RB: GO TO
    8040
8030 LET WB=WB-RB
8040 IF RW=1 THEN  PLOT OVER
    1;RDX,175-RDY
8050 LET RW=0: IF ABS WB>57
    THEN  RETURN
8060 LET RDX=X3+INT (((X4-X3
    )/2)-SIN (WB*(PI/180))
    *(X4-X3)*6)
8070 LET RDY=Y3+((Y4-Y3)*((
    RDX-X3)/(X4-X3))
    +2)
8080 IF RDY<2 OR RDY>90
    OR
    RDX<11 OR RDX>244 THEN
    RETURN
8090 LET RW=1: PLOT  OVER 1;
    RDX,175-RDY
8100 RETURN
```

El POKE de la línea 1 coloca al ordenador en modo de letras mayúsculas. La línea 5 pone a 0 todas las variables. La línea 110, que introdujiste en la primera parte, envía el programa a la línea 5000 y dibuja la cabina del aeroplano; a continuación la línea 5080 define las variables que controlan la posición del avión en el cielo: GZ se refiere a la distancia que se mide a lo largo del eje Z, es decir su altura hacia arriba o hacia abajo; VZ es la velocidad según este mismo eje; VY se refiere a la velocidad del avión hacia adelante o hacia atrás, es decir

a lo largo del eje Y; WY es la velocidad del viento en la misma dirección. GX, VX y WX corresponden respectivamente a la distancia, velocidad y velocidad del viento en la direcciónе del eje X, es decir de izquierda a derecha.

La línea 5090 corresponde al cuentarrevoluciones y la 5100 define los valores límite para el balance (RT), cabezada (TP) y anchura de la pista de aterrizaje (WR).

La línea 5500 es el comando temporal para la rutina principal de control, que introducirás en la próxima parte de nuestro coleccionable.

La línea 5510 te envía a la subrutina que empieza en la línea 1000 y termina en la línea 1110. Esta subrutina actualiza todas las variables durante el vuelo del avión. Las líneas 1025 y 1030 sirven para comprobar si has permitido que tu avión entre en pérdida de velocidad, dejando que su velocidad caiga por debajo de los 30 metros por segundo. Caso de que ocurra esto, te envía a la subrutina de las líneas 1500 a 1520, que reproduce el efecto de una pérdida de velocidad. La línea 1510 sirve para convertir en incierto el resultado de lo que sucederá cuando el avión entre en pérdida de velocidad; puede ser que caigas en línea recta o que empieces a dar vueltas a medida que vas cayendo.

La subrutina a la que te envía la línea 1100, contenida en las líneas 500 y 510, calcula el ángulo con el que estás volando.

La siguiente subrutina importante, que empieza en la línea 2180 y termina en la 2510, dibuja de nuevo los diales y los contadores cuando se actualiza la información que presentan. La línea 2180 comprueba la imagen del contador GC para ver si hay una imagen que tiene que ser sustituída por otra. La subrutina de la línea 2200 dibuja la nueva posición de la manecilla del *dial* de la velocidad del aire. Las líneas 2205 y 2210 calculan y dibujan la nueva posición de las dos manecillas del *dial* de altitud. La línea 2230 desplaza la manecilla del nuevo contador, con arreglo a los cálculos hechos en las líneas 2225 y 2226.

El cuentarrevoluciones se actualiza en la línea 2240. Las líneas 2250 a 2270 calculan la nueva demora de la pista de aterrizaje y la deriva, mientras que la línea 2280 calcula y presenta la distancia.

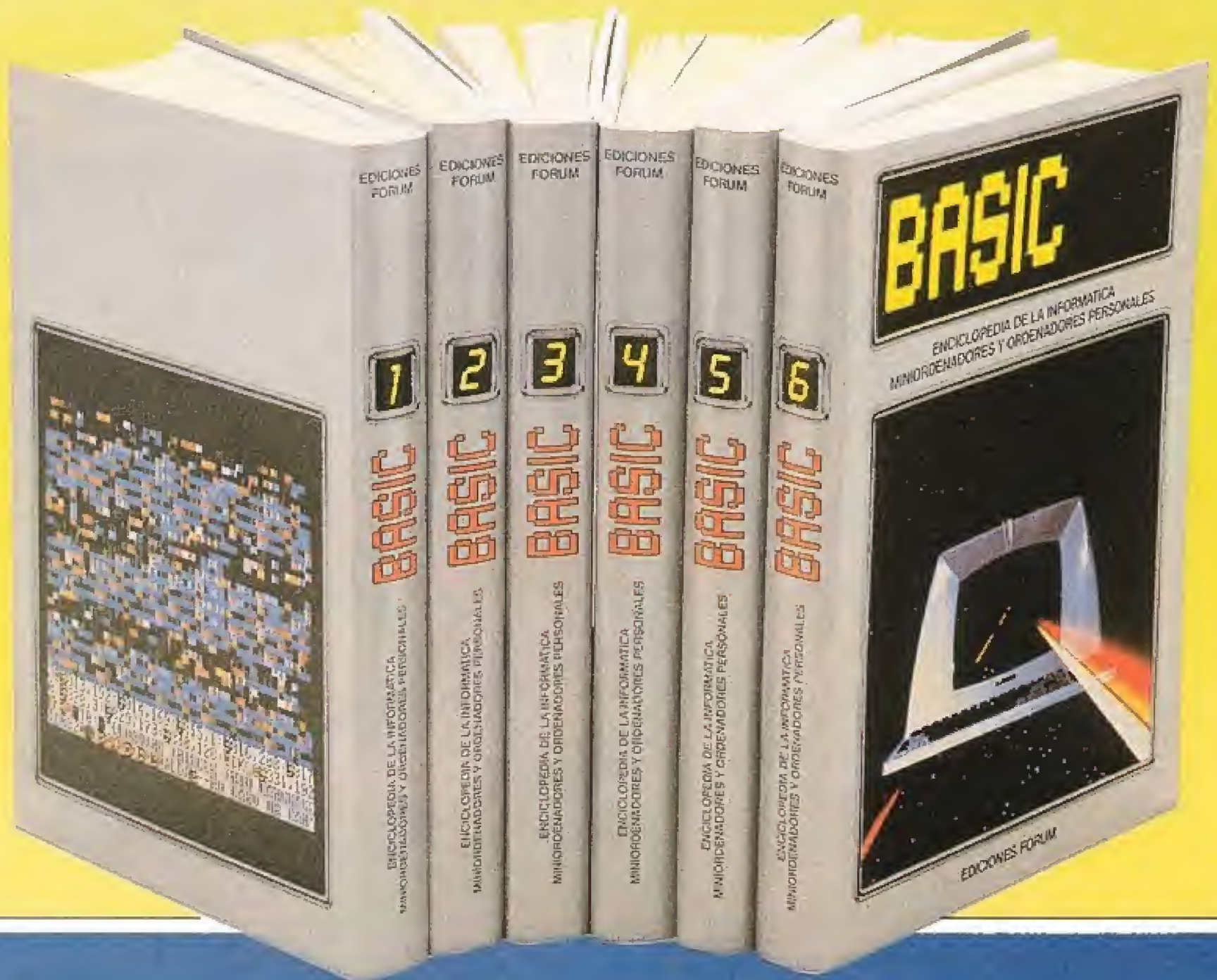
Las líneas 2280 a 2370 se ocupan de calcular y dibujar el horizonte artificial del segundo *dial*.

En la 2376 se comprueba la línea del horizonte real y las 2377 a 2445 lo calculan y dibujan de nuevo cuando es visible a través de la ventanilla de la cabina.

La línea 2500 te envía a la subrutina que comienza en la línea 8000 y termina en la 8100, la cual se ocupa de calcular y dibujar la imagen radar de la pista de aterrizaje, que es visible en la parte superior de tu pantalla, así como el punto de dicha pista que aparece justo por debajo de la línea del horizonte a través de la ventanilla de la cabina, en los casos en que es visible.

BASIC

ENCICLOPEDIA DE LA INFORMATICA DE LOS MINIORDENADORES Y ORDENADORES PERSONALES



84 fascículos semanales de 24 páginas cada uno.

6 volúmenes de gran formato (19,5 × 27,5) encuadernados en geltex impreso a todo color.

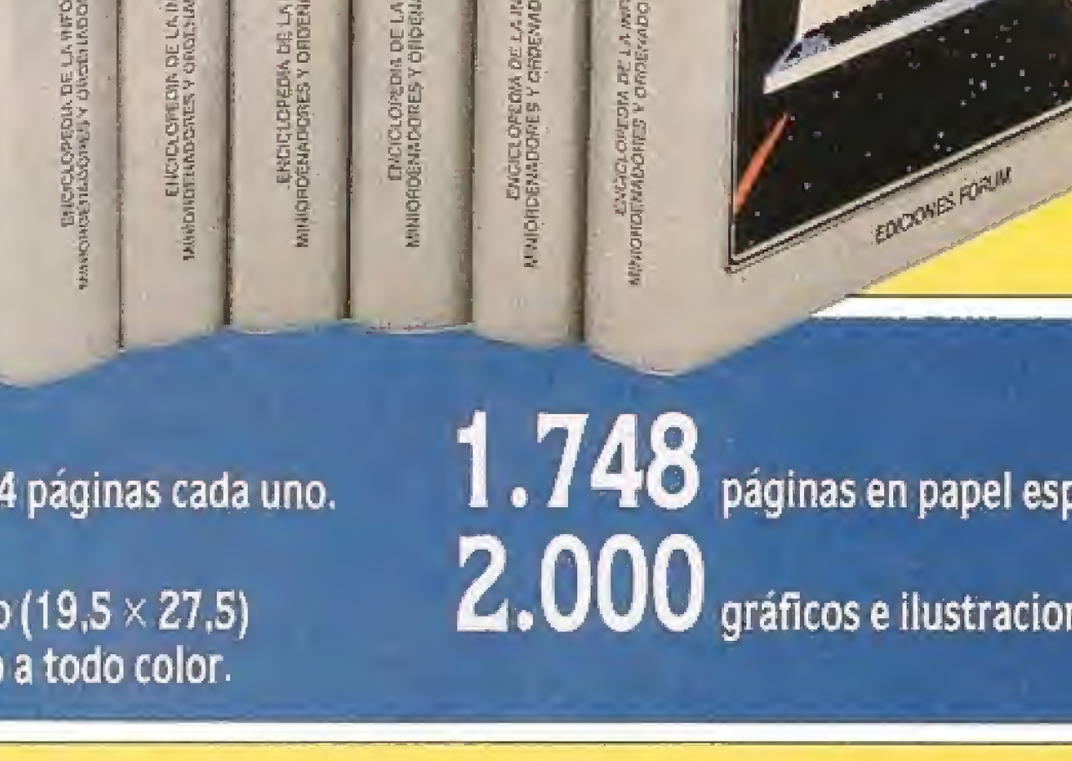
1.748 páginas en papel especial.

2.000 gráficos e ilustraciones a color.

BASIC
Una información indispensable del primero al último fascículo.

BASIC
Para no ser un extraño en el futuro tecnológico que nos aguarda.

BASIC
Para poner una nueva ciencia a nuestro servicio.



SUPER OFERTA DE LANZAMIENTO

RECORTE ESTE CUPON Y ENVIELO A EDISA (Dpto. de Suscripciones), López de Hoyos, 141 - 28002 Madrid

SI, deseo suscribirme a **BASIC** y recibiré en mi hogar 4 fascículos al mes, abonando sólo **700 Ptas.** por cada envío. El servicio a mi domicilio es GRATUITO.

Con su primer fascículo recibirá GRATIS el fascículo n.º 2, es decir, su primer envío constará de 5 fascículos al precio de 4.

▽ POR FAVOR, RELLENE SUS DATOS EN MAYUSCULAS ▽

NOMBRE _____
APELLIDOS _____
DIRECCION _____ N.º _____
PISO _____ COD. POSTAL _____
CIUDAD _____
PROVINCIA _____
TLFNO. _____
FIRMA _____

AGREGALE INSTRUCCIONES A TU BASIC

El BASIC de tu ordenador no es algo cerrado y fijo. Es un programa como otro cualquiera que tú puedes adaptar a la medida de tus necesidades añadiéndole nuevas instrucciones.

El BASIC es precisamente un conjunto de programas en lenguaje máquina que corren en tu ordenador. Aunque las instrucciones que ejecuta el procesador realmente llegan desde la ROM (no desde el teclado la cinta o el disco), en la mayoría de los micros es posible adentrarse en el BASIC, incluso llegando a añadir o suprimir comandos.

La adición de instrucciones puede ser muy útil si estás utilizando tu ordenador para alguna aplicación muy específica que requiera repetidamente rutinas complejas. En este artículo introduciremos un par de instrucciones para el BASIC del **Spectrum**, que podrás utilizar si posees la **Interface 1** y 48K de memoria.

La adición de una instrucción al BASIC ha de hacerse mediante dos operaciones. En primer lugar hay que agregar una rutina al editor del BASIC para que reconozca la sintaxis de la nueva instrucción, ya que en caso contrario cuando se encuentre con ella enviará un mensaje de error. A continuación hay que agregar otra rutina al cuerpo principal del BASIC para ejecutar la nueva instrucción cuando se haga correr el programa.

En la siguiente rutina se añaden las instrucciones INVERSE y ATTR. Observa que estas palabras reservadas ya existen en el **Spectrum**, pero normalmente INVERSE ha de ir seguido por un 0 o un 1. Con el 0 todo permanece en el modo normal, mientras que con el 1 se invierten todas las salidas a pantalla a partir de ese punto. El nuevo INVERSE que añadimos aquí no requiere parámetros e invierte toda la pantalla.

El ATTR del **Spectrum** es normalmente una función y no una instrucción. Lo utilizas para asegurarte del atributo de una dirección de pantalla especificada por dos parámetros entre paréntesis que figuran a continuación de la palabra reservada. El nuevo ATTR es un comando. Va acompañado de un parámetro (no figura entre paréntesis) que especifica el atributo de toda la pantalla.

Recuerda que esta rutina sólo te funcionará si tienes conectada la **Interface 1**.

Teclea para Spectrum

```
10 REM ORG 65200
20 REM RST 16
30 REM DEFW $0018
40 REM CP 221
50 REM JR Z,INV
60 REM CP 171
70 REM JR Z,ATTR
80 REM JP $01F0
90 REM INV RST 16
100 REM DEFW $0020
110 REM CALL $05B7
120 REM LD HL,$4000
130 REM LD B,24
140 REM INVLO PUSH BC
150 REM LD B,0
160 REM INVLI LD A,(HL)
170 REM CPL
180 REM LD (HL),A
190 REM INC HL
200 REM DJNZ INVLI
210 REM POP BC
220 REM DJNZ INVLO
230 REM JP $05C1
240 REM ATTR RST 16
250 REM DEFW $0020
260 REM RST 16
270 REM DEFW $1C82
280 REM CALL $05B7
290 REM RST 16
300 REM DEFW $1E94
310 REM LD HL,22528
320 REM LD (HL),A
330 REM LD DE,22529
340 REM LD BC,767
350 REM LDIR
360 REM RRCA
370 REM RRCA
```

- NUEVAS INSTRUCCIONES
- CAMBIO DE UNA ROM A OTRA
- NUEVAS PALABRAS
- COMPROBACION DE LA SINTAXIS
- TRATAMIENTO DE LOS ERRORES

```
380 REM RRCA
390 REM OUT 254,A
400 REM JP $05C1
410 REM END
900 CLEAR 65199
```

El origen de esta rutina está en 65200. Pero no se la llama de la forma acostumbrada. No querrás estar ejecutándola a todas horas, ya que en ese caso la rutina que hace correr el BASIC no podría funcionar. En lugar de eso, se infiltra entre el BASIC y los mensajes de error.

Cuando está funcionando el editor del BASIC y se encuentra con que no reconoce la instrucción o la sintaxis de la línea activa, comprueba a la variable VECTOR del sistema. Normalmente esta variable le encamina hacia las rutinas de mensajes de error que hay en $01F0. Pero si en VECTOR se ha POKEado la dirección de comienzo de otra rutina, como ocurre en este caso, el procesador se irá a ejecutar esta rutina.

CAMBIO DE UNA ROM A OTRA

La ROM que lleva incorporada la propia **Interface 1** añade nuevas ins-

trucciones al BASIC. Principalmente están relacionadas con el **Microdrive** y el manejo de sus instrucciones SAVE y LOAD. Pero también añade otras instrucciones tales como CLS1, que borra la pantalla y retorna a los colores iniciales.

Al poner la **Interface 1** se hace necesario no obstante cambiar a la nueva ROM del *interface* desde la ROM vieja que lleva el **Spectrum**. Normalmente el procesador atiende a lo que hay en la nueva ROM del **Interface 1**, pero la instrucción **rst 16**, es decir recomenzar en 16 lo envía a la nueva rutina de inicialización que hay en $0010, la cual a su vez le dirige a la ROM antigua. Pero la rutina de recomenzar no se limita a enviar al procesador a cualquier sitio antiguo de la ROM vieja. Lo envía específicamente a la rutina que empieza en la dirección contenida en los dos bytes que siguen a la instrucción **rst**.

En consecuencia **rst 16**, **defw $0018** fuerzan un salto a la rutina situada en la dirección $0018 de la vieja ROM. Esta rutina hace lo que haría **rst 24** si no se hubiera conectado el **Interface 1**. Permite obtener el código del comando que aparece al principio de la siguiente instrucción del BASIC.

La instrucción **cp 221** compara el código del comando con el código de INVERSE. Si ambos códigos coinciden, la instrucción **jr z, inv** envía al procesador a la rutina **inv**.

Si no coinciden los dos códigos, la instrucción **cp 171** compara el código del comando con el de ATTR. Si ahora sí hay coincidencia, **jr z,attr** envía al procesador a ejecutar la rutina **attr**. Si tampoco coinciden estos dos códigos, **jp$01F0** envía al procesador a la rutina de sintaxis de error situada en 496, que es el sitio a donde habría ido en primer lugar si esta rutina no hubiese interceptado su camino.

LA RUTINA «INVERSE»

La instrucción **rst 16** conmuta de nuevo la ROM antigua y llama a la rutina que hay en $0020, la cual mueve el puntero al siguiente byte de la línea de BASIC.

A continuación se llama mediante call a la rutina situada en $05B7. Esta rutina sirve para comprobar si hay una marca de final de sentencia. Si encuentra una durante el tiempo de sintáxis, es decir, cuando el editor está trabajando sobre una línea prefijada con un número de línea, queda en condiciones de aceptar la siguiente línea o un comando en modo directo.

Cuando el ordenador está en tiempo de ejecución, es decir, cuando se está introduciendo un comando en modo directo o se está haciendo correr un programa, esta rutina envía al procesador de nuevo al principio para ejecutar la siguiente instrucción.

La instrucción **ld hl,$4000** carga el registro HL con la primera dirección de pantalla. El registro B se utiliza entonces como un contador doble. En primer lugar se carga el número 24 en B y se mete en el *stack*; a continuación se carga el 0 (que actuará como 256 cuando empieces a decrementar). Existen 24×256 posiciones de pantalla.

La instrucción **ld a,(hl)** carga en el acumulador el contenido de la primera posición de pantalla, es decir la correspondiente a la esquina superior izquierda. Con **cpl** se obtiene el complemento de dicho valor, convirtiendo todos los ceros en unos y los unos en ceros. Seguidamente **ld (hl)**,a sitúa el resultado de esta operación en la misma posición de pantalla.

El resultado es que se activan todos los *pixels* que estaban desactivados y se desactivan los que estaban activados. En otras palabras, se invierte esa parte de la pantalla.

CONTANDO A TRAVES DE LA PANTALLA

A continuación se incrementa el registro HL para desplazarse a la siguiente posición de la pantalla. La ins-

trucción **djnz invli** decrementa el registro B y da un salto hacia atrás a **invli** siempre que la cuenta no haya llegado hasta cero. De esta forma el bucle se ejecuta 256 veces, moviéndose cada vez hacia la siguiente posición e invirtiéndola. Cuando la cuenta llega a cero, se saca el otro contador del *stack*, se decrementa y, si su cuenta no es cero, el procesador salta a **invlo** que hace que el contador sea de nuevo puesto en el *stack*.

Este bucle externo se ejecuta 24 veces mientras que el interior se ejecuta 256 veces cada vez que el procesador recorre el externo, con lo cual resulta un total de 6144 veces. Cuando llega a cero la cuenta del contador del bucle externo, ya han sido invertidas todas las direcciones de pantalla y **jp $05C1** envía al procesador a la rutina que comienza en $05C1 la cual le dirige hacia la siguiente sentencia del BASIC para que empiece a ejecutarla.

LA RUTINA «ATTRIBUTE»

Esta rutina empieza exactamente de la misma manera que la rutina INVERSE, enviando al procesador a la rutina de la ROM antigua encargada de desplazar el puntero al siguiente byte de la palabra reservada.

Sin embargo en esta ocasión tiene que haber un parámetro. Por eso **rst 16** y **defw $1C82** envían al procesador a la rutina de la ROM antigua que se encarga de evaluar las expresiones numéricas. El resultado se pone en el *stack*.

De nuevo se llama a la rutina $05B7. Es la rutina que comprueba el final de sentencia cuya salida se produce en tiempo de sintaxis. Si a continuación de la expresión numérica no hay una marca de fin de sentencia sino un signo de puntuación, una letra o una nueva expresión numérica, la rutina dará un mensaje de error. El nuevo comando sólo requiere un parámetro, por lo que si detrás hay algo más, la sintaxis está equivocada.

El **Spectrum** no se sentirá confundido si utilizas la función normal ATTR que lleva dos parámetros. Esto ya ha sido tenido anteriormente en cuenta por las propias rutinas de BASIC del **Spectrum**. De producirse, habría sido aprobado e introducido y el procesador no llegaría a las rutinas de los mensajes de error donde se intercepta el programa. Así se aceptan tanto la función INVERSE antigua como el nuevo comando INVERSE, así como los dos ATTR.

Las instrucciones **rst 16** y **defw $1594** envían al procesador a la rutina de la ROM antigua que se encarga de obtener el valor del parámetro del *stack*. Dicho valor queda depositado en el acumulador.

La instrucción **ld hl,22528** carga ahora la dirección del principio del fichero de atributos en el registro HL; **ld (hl),a** carga el valor numérico del parámetro INVERSE en la primera dirección del fichero de atributos.

Con **ld bc,767** se carga el registro BC con el valor 767 para ser utilizado como contador. El fichero de atributos contiene 768 posiciones, organizadas en 24 filas y 32 columnas, pero una de dichas posiciones ya ha sido procesada. El bloque **ldir** carga el contenido de la dirección a la que apunta HL en la dirección contenida en DE, después incrementa HL, y DE , decrementa BC y repite el proceso si el resultado no es cero.

En otras palabras, lo que hace es copiar el atributo de la primera posición de pantalla, la que acaba de tratar el programa, en los atributos de todas las demás posiciones de pantalla.

La siguiente cosa que hay que hacer es cambiar el color del borde para que coincida con el color del papel. Para hacer esto se utiliza el comando **out**. El problema es que el color del papel se encuentra almacenado en los bits 3, 4 y 5 de los atributos, mientras que el color del borde tiene que estar en los bits 0, 1 y 2 con el comando **out**.

Por esta razón las tres instrucciones **rrca** empujan el valor del atributo, que todavía se encuentra en el acumu-

lador. tres lugares hacia la derecha. A continuación se produce el **out** por el *port* 254.

Una vez que se ha hecho todo esto, **jp $05C1** envía de nuevo al procesador a la rutina que hay en $05C1.

EJECUCION DEL PROGRAMA

El comando CALL situado al final del programa desplaza automáticamente el límite superior del BASIC para proteger el código máquina. Para conseguir que este programa ejecute cada vez uno de los nuevos comandos que se le introducen, la variable VECTOR debe contener la dirección de comienzo de esta rutina.

Por eso tienes que poner dicho valor en el puntero de dos bytes con ayuda de los siguientes POKEs:

```
POKE 23735,176:
POKE 23736,254
```

Ten mucho cuidado al teclear los anteriores POKEs. Si te equivocas al introducir cualquiera de ellos y cometes algún error de sintaxis, el procesador se dirigirá a un lugar equivocado y el sistema se quedará inútil.

Todavía es más crítico que introduzcas correctamente el segundo POKE. El primer POKE cambia la mitad de la variable VECTOR. Por ello, si se produce un error de sintaxis en la segunda mitad, el procesador será mal encaminado con resultados desastrosos.

INPUT sinclair

SERVICIO DE EJEMPLARES ATRASADOS

¡NO TE PIERDAS NI UN SOLO EJEMPLAR!

INPUT SINCLAIR quiere proporcionar a sus lectores este nuevo servicio de ejemplares atrasados para que no pierdan la oportunidad de tener en sus hogares todos los ejemplares de esta revista, líder en el mercado español.

Podréis solicitar cualquier número de **INPUT SINCLAIR** que querais, siempre al precio de cubierta (sin más gastos).

Utiliza el cupón adjunto, enviándolo a **EDISA** (Dpto. de Suscripciones), López de Hoyos, 141 - 28002 Madrid, o bien llámanos por teléfono al (91) 415 97 12.

INPUT sinclair

siempre a tu servicio

CUPON DE PEDIDO

SI, envíenme contrarreembolso ..... ejemplares de **INPUT SINCLAIR** de los números:

(marca con una (X) tu elección)

1	2	3	4	5	6	7	8	9	10	11	12

NOMBRE

APELLIDOS

DOMICILIO

NUM. PISO ESCALERA COD. POSTAL

POBLACION PROV.

TELEFONO FIRMA

Código máquina

NUMEROS BAJO CERO

- CUANDO LOS NUMEROS NEGATIVOS SE HACEN NECESARIOS
- PROGRAMA DE CONVERSION PARA NUMEROS NEGATIVOS
- LA CONVENCION DEL SIGNO

La comprensión de los números binarios y hexadecimales no suele ser difícil, pero la expresión de valores negativos con estos sistemas de numeración puede presentar problemas.

En la programación de algunos juegos puede ser necesario utilizar valores negativos, por ejemplo, para desplazarse sobre la pantalla en determinadas direcciones. Las direcciones van codificadas en grupos de ocho bits, ya que esa es la única manera en que el ordenador puede almacenar los datos en la memoria. Pero surge un problema: como hemos visto, un byte puede representar cualquier número desde 0 hasta 255, es decir, desde 0000 0000 hasta 1111 1111. Pero con esto se agotan las posibilidades de un número binario de ocho bits, que ya no tiene sitio para un signo menos o más, ni forma alguna de representarlo.

En la aritmética ordinaria, al restar 1 de 0 se obtiene el valor −1; aquí tienes ese mismo cálculo realizado con números binarios de ocho bits:

```
0000 0000
- 1
-----------
1111 1111
```

Puedes probar a tomar un 1 prestado de la columna situada más a la izquierda, pero cuando el número binario está limitado a ocho bits no hay manera de hacer nada. Análogamente, al restar otro 1 (en la aritmética tradicional se obtendría como resultado un −2) tenemos 1111 1110, que en binario corresponde al número decimal 254, ya que 1111 1111 es 255.

Esto no son curiosidades típicas de los números binarios: imagínate por ejemplo lo que ocurriría en decimal si sólo hubiese tres sitios o columnas en las que poner los números. Mira lo que sucedería si se intentase sumar 100 a 999 con estas limitaciones:

```
  100
+ 999
-------
(1)099
```

El uno que figura en la columna de las unidades de mil, está escrito entre paréntesis; en un sistema aritmético con sólo tres columnas no hay espacio para las que «te llevas». En un siste-

ma de sólo tres lugares, el resultado de esta suma es pues 99, o sea, exactamente el resultado de 100 menos 1.

De la misma forma, sumar 98 con 100 en un sistema de tres lugares equivale a restar 2, mientras que restar 998 de 100 equivale a sumarle 2.

En conclusión, una misma sucesión de números de un byte puede representar un número positivo o negativo, sin tener en cuenta la posible confusión o dificultad en el cálculo.

¿Qué hacer entonces? En la mayoría de las aplicaciones de los ordenadores domésticos, este problema no suele plantearse: tanto las direcciones de memoria como los códigos de operación, ambos expresados en binario, pueden considerarse siempre como positivos. Las únicas ocasiones en que se encuentran números negativos son en los datos o en los saltos del código máquina. equivalentes a los GOTOs.

CAMBIANDO LOS BITS

Para pasar de un valor positivo en binario a su correspondiente negativo, hay que aplicar el llamado procedimiento del complemento a 2; no tiene una base teórica de fácil comprensión, pero sin embargo funciona bien.

Para pasar desde un número en binario a su correspondiente valor negativo, hay que cambiar los ceros por unos y viceversa, sumando al final 1 al resultado.

El siguiente programa realiza precisamente esta función. Observa que cuando el número en binario está limitado a ocho cifras, su equivalente en hexadecimal consta exactamente de dos cifras: esto quiere decir que también el equivalente hexadecimal del complemento a 2 es el correspondiente valor negativo.

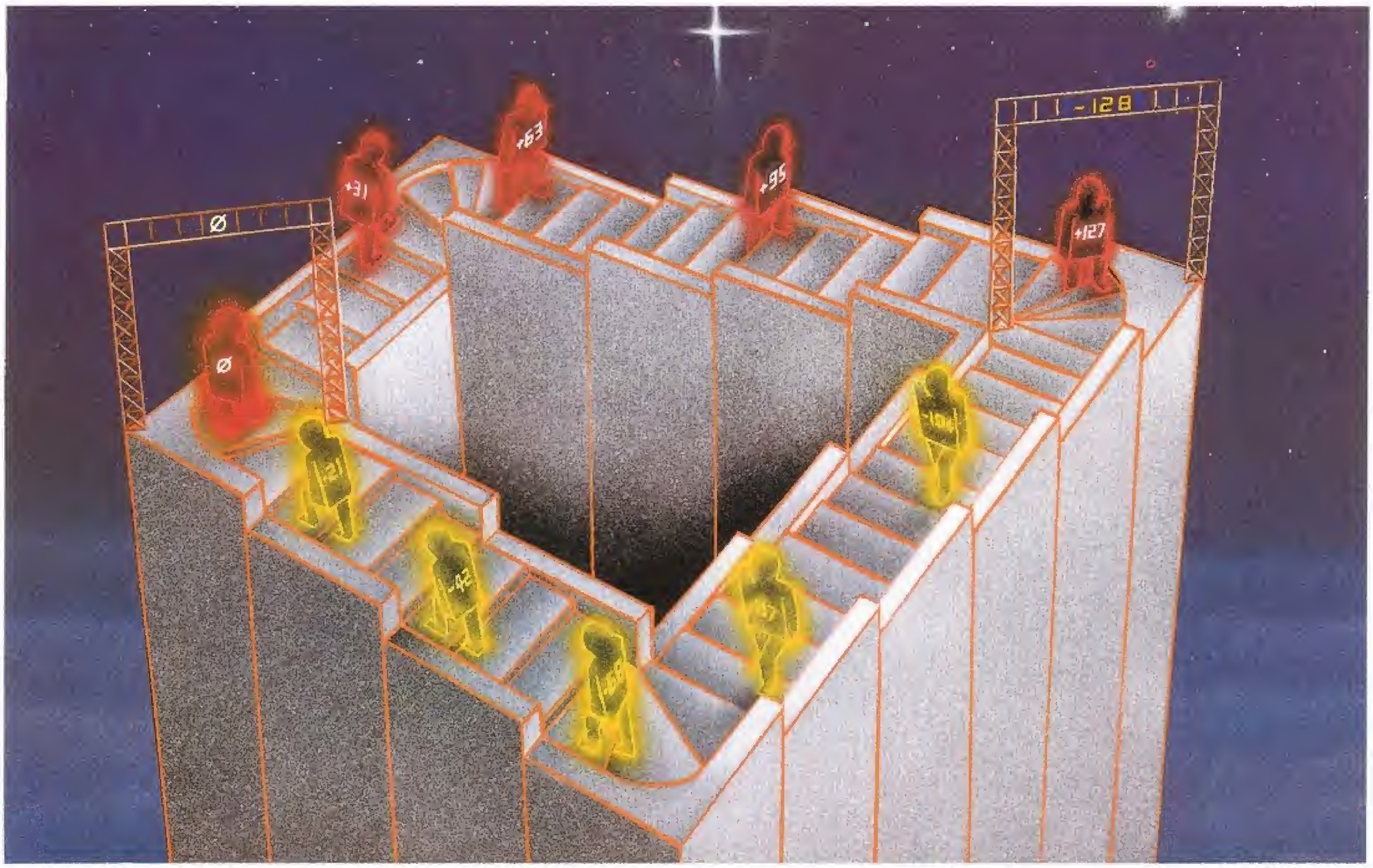
Teclea

```
10 PRINT AT 0,7;"NUMERO
   NEGATIVO"
20 PRINT AT 2,1;"DEC";TAB
   14;"BIN";TAB 28;"HEX"
30 LET A$="[19*ESPACIO]"
40 FOR N=7 TO 13
50 PRINT AT N,6; INVERSE 1;
   A$
60 NEXT N
70 PRINT AT 8,8;
   "COMPLEMENTO"
80 PRINT AT 10,3;"+";AT 10,
   30;"+"
90 PRINT AT 15,8;
   "COMPLEMENTO A 2"
95 LET C=0
100 LET DD=-C: DIM A(8)
110 PRINT AT 4,0;"[4*
    ESPACIO]";AT 4,4-LEN
    STR$ C;C
115 POKE 23608,C: LET E=PEEK
    23608: LET Z=E: GO SUB
    300: PRINT AT 4,29;A$
120 PRINT AT 17,0;"[4*
    ESPACIO]";AT 17,4-LEN
    STR$ DD;DD
130 LET D=128: LET CC=E
```

GANADORES DE LOS MEJORES DE INPUT SINCLAIR

En el sorteo correspondiente al número 12 entre quienes escribisteis mandando vuestros votos a LOS MEJORES DE INPUT han resultado ganadores:

NOMBRE	LOCALIDAD	JUEGO ELEGIDO
Francisco Guirado Molina	La Cabrera (Madrid)	Skyfox
J. Jorge Prunés Casterás	Terrassa (Barcelona)	Equinox
Rodrigo Miranda De Sasia	Madrid	Quazadron
Camilo Rodríguez Sánchez	Vigo (Pontevedra)	Kunf fu Master
J. Jaume Freixes Crusat	El Masroig (Tarragona)	Phantomas II
Charo Oñate Cabanillas	Bilbao	Skyfox
F. Javier Wisental Gómez	Sevilla	Spindizzy
Rafael García Ruiz	Aranda de Duero (Burgos)	Green Beret
Eugenio Vives Lamarga	Reus (Tarragona)	Sgrizan
G. Adolfo Pacheco Navas	Barcelona	Winter Games




```
140 FOR N=1 TO 8: LET A(N)=0
150 IF CC-D>=0 THEN LET A(N)
    =1: LET CC=CC-D
160 PRINT  BRIGHT 1;AT 4,6+2
    *N;A(N);AT 10,6+2*N;1-A
    (N)
170 LET D=D/2: NEXT N
180 POKE 23608,DD: LET DD=
    PEEK 23608: LET D=128
185 LET Z=DD: GO SUB 300:
    PRINT AT 17,29;A$
190 FOR N=1 TO 8: LET B=0:
    IF DD-D>=0 THEN LET B=1:
    LET DD=DD-D
200 PRINT  BRIGHT 1;AT 17,6+
    2*N;B: LET D=D/2: NEXT N
210 PRINT AT 18,0;"----  ---
    ---------------  ----"
220 PRINT AT 19,3;"0[4*
    ESPACIO]0 0 0 0 0 0 0 0
    [6*ESPACIO]00"
230 IF INKEY$="" THEN  GO TO
    230
240 LET A$=INKEY$: IF A$=" "
    THEN  LET C=C+1: IF C=
    128 THEN  LET C=-128:
    BEEP 1,1
250 IF A$="B" OR A$="B" THEN
    LET C=C-1: IF C=-129
    THEN  LET C=127: BEEP
    1,1
260 IF A$<>" " AND A$<>"B"
    AND A$<>"B" THEN INPUT "
    ?";C
270 GO TO 100
300 LET ZA=INT (Z/16): LET
    ZB=Z-(16*ZA)
310 LET ZA=ZA+48: IF ZA>57
    THEN  LET ZA=ZA+7
320 LET ZB=ZB+48: IF ZB>57
    THEN  LET ZB=ZB+7
330 LET A$=CHR$ ZA: LET A$=
    A$+CHR$ ZB: RETURN
```

LA CONVENCION DEL SIGNO

Como hemos dicho, para muchos fines un número en binario o en hexadecimal corresponde perfectamente a un número positivo o negativo. Sin embargo hay ocasiones en que se quiere conocer si un número es positivo o negativo.

Los saltos realizados en los programas en lenguaje máquina requieren que se especifique el número de bytes que abarcan, en valores positivos si el salto es hacia adelante y negativos si el salto es hacia atrás. El ordenador examina el primer bit del número binario y determina por sí solo si se trata de un número positivo o negativo. Si el primer bit es un 1, el ordenador considera el número como negativo, mientras que si es 0 lo toma como positivo.

Este procedimiento se ajusta a una determinada convención de signos. Esto significa que el ordenador, en vez de manejar números binarios que varían entre 0 y 255, considera que su margen de variación va desde −128 a +127. En el programa de conversión en complemento a 2, habrás oído la señal de «bip» al llegar a 128. Esto se debe a que 128, que en binario es 1000 0000 y en hexadecimal es $80, corresponde a un valor negativo. (Haz la prueba: −1000 0000 + 1000 0000 = (1)0000 0000, que es 0 en binario de ocho bits; 80 + 80 = (1)00, que es 0 en hexadecimal de dos cifras; 128 − 128 = 0 en decimal.)

¿Qué sucede entonces? Dado que 1000 0000 tiene un 1 en su primer bit, el ordenador lo considera como un número negativo, el −128. Por otra parte el cero, que es 0000 0000, tiene un 0 en su primer bit, por lo que el ordenador lo toma como un número positivo.

¡PARTICIPA EN INPUT!

BASES

PROGRAMAS: Una vez desarrollado tu programa, que debe ser original y no haber sido enviado a ninguna otra publicación, puedes enviárnoslo aquí grabado en cassette, diskette o microdrive. Es preferible que vaya acompañado por un listado de impresora, pero no es imprescindible.

El programa habrá de venir acompañado por un texto que aclare cuál es su objetivo, el modo de funcionamiento y una explicación del cometido que cumplen las distintas rutinas que lo componen. El texto se presentará en papel de tamaño folio y mecanografiado a dos espacios. No importa que la redacción no sea muy clara y cuidada; nuestro equipo de expertos se encargará de proporcionarle la forma más atractiva posible.

ARTICULOS E IDEAS: Se aplica lo anteriormente dicho para los textos que acompañan a los programas; es decir, conviene detallar al máximo lo que desees que aparezca publicado en la revista, de la manera que te gustaría que otra persona hubiera explicado eso mismo.

UN JURADO propio decidirá en cada momento qué colaboraciones reúnen los requisitos adecuados para su publicación, y evaluará la cuantía del premio en metálico al que se hagan acreedoras.

No olvidéis indicar claramente para qué ordenador está preparado el material, así como vuestro nombre y dirección y, cuando sea posible, un teléfono de contacto. Entre todos los trabajos recibidos durante cada mes SORTEAREMOS:

- **Un premio de 50.000 ptas.**
- **Un premio de 25.000 ptas.**
- **Un premio de 10.000 ptas. en material microinformático a elegir por los afortunados.**

¡No os desaniméis!, por muy simples o complejas que puedan parecer vuestras ideas, todas serán revisadas con el máximo interés.

INPUT SINCLAIR
Alberto Alcocer, 46, 4.º B
28016 Madrid

NOTA: INPUT no se responsabiliza de la devolución del material que no vaya acompañado por un sobre adecuado con el franqueo correspondiente.

MANEJO DE PANTALLA DESDE C/M

Quizá uno de los temas más vistosos e interesantes del código máquina sea el manejo de la pantalla. Disponemos de una serie de instrucciones sencillas y muy potentes que nos permiten trabajar con el área de la memoria utilizada para la imagen.

Dentro de esta gama de instrucciones podemos incluir las del grupo de LDIR y las rotaciones y desplazamientos como las RRA, SRL... aunque no son los únicos recursos.

UTILIZANDO LAS ROTACIONES Y DESPLAZAMIENTOS

Con este tipo de instrucciones podemos conseguir mayor control sobre cada bit (un punto de la pantalla) lo que nos da mayor versatilidad de operación, más incluso de la que disponemos desde el BASIC.

* **Rotaciones**: Principalmente consisten en el «giro» de los bits del byte contenido en un registro (normalmente el A), o en una dirección de memoria apuntada por un registro doble, como el HL. Proponemos como ejemplo A y HL pero sirven los demás)

- RLCA o RLC (HL) (*Rotate Left Circular*): «Giro» de los bits hacia la izquierda, en el que el bit 0 pasa al lugar del 1, el bit 1 al del 2, y así hasta el séptimo bit que pasa al lugar del bit 0, que además pasa al indicador de acarreo (*carry flag*) que puede almacenar un bit, se pone a 1 ó 0 según fuera el valor del bit 7 (Figura 1).

- RRCA o RRC (HL) (*Rotate Right Circular*): Exactamente igual que el caso anterior pero el giro se produce a la derecha, y es el valor del bit 0 el que además de pasar al 7, es almacenado en el acarreo (Figura 2).

- RLA o RL (HL) (*Rotate Left*): Se produce un giro a la izquierda pero el valor que pasa al lugar del bit cero es el valor acumulado en el acarreo. El contenido del bit 7 pasa al acarreo (Figura 3).

- RRA o RR (HL) (*Rotate Right*): Parecido al anterior pero con el giro a la derecha (Figura 4).

- Sólo recordar la existencia de las instrucciones RLD y RRD que no afectan al acarreo e implican el intercambio de grupos de 4 bits entre el registro A y la dirección apuntada por el (HL). Son más complicadas de utilizar y no nos interesa tanto su uso.

* **Desplazamientos**: De estructura similar a las anteriores pero sin darse el intercambio entre los bits de los extremos.

- SLA r (*Shift Left Arithmetic*): Todos los bits del registro «r» ocupan el lugar del de la izquierda, poniéndose a cero el primer bit (el número 0), y pasando el contenido del séptimo bit al acarreo (Figura 5).

Además el movimiento provocado por esta instrucción produce la multiplicación por 2 del valor contenido en el registro (Tabla 1):

- SRA r (*Shift Right Arithmetic*): Desplazamiento de los bits del registro «r» a la derecha, quedando inalterado el número 7, y pasando el bit 0 al acarreo (Figura 6).

FIGURA N.° 1

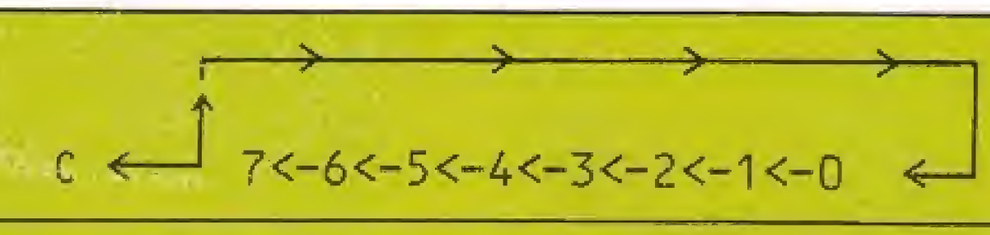


FIGURA N.° 2

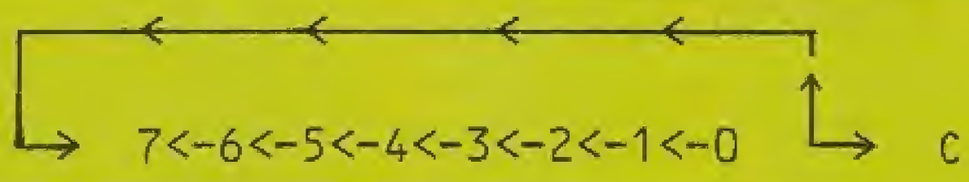


FIGURA N.° 3

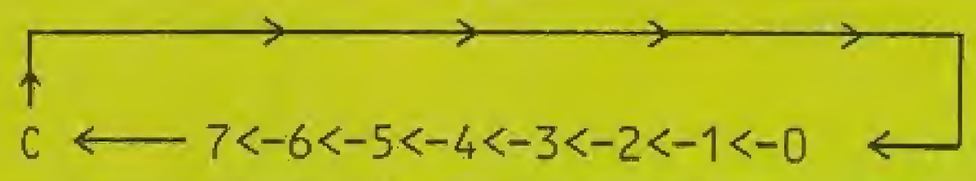


FIGURA N.° 4

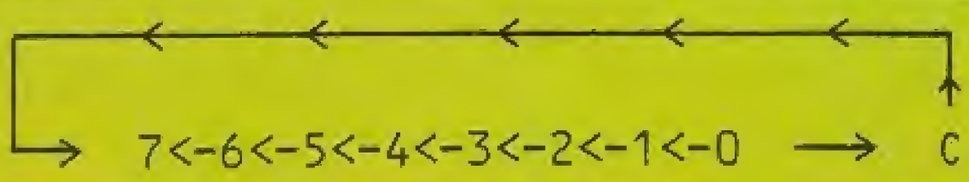


FIGURA N.° 5

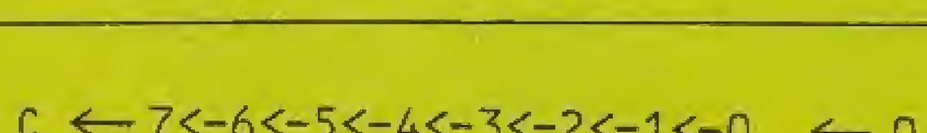


FIGURA N.° 6

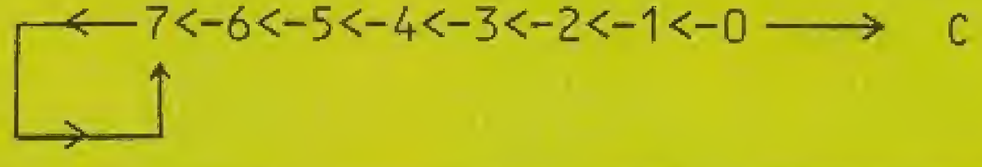


FIGURA N.° 7

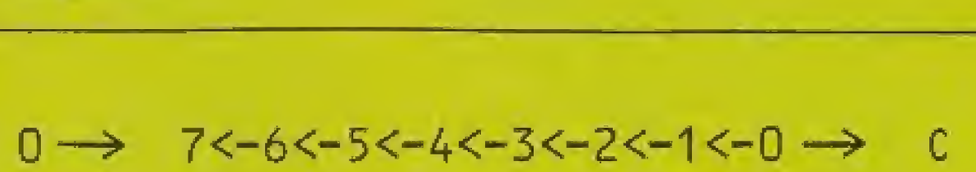


TABLA 1

```
Decimal-->       24 --> SLA -->       48
Binario--> 00011000 --> SLA --> 00110000
```

- SRL r (*Shift Right Logical*): Similar a SLA r pero con «giro» a la derecha (Figura 7).

A continuación adjuntamos una sencilla rutina en C.M. que nos sirve para ir probando varias de las instrucciones vistas anteriormente (Ver listado Rutina 1).

RUTINA 1

```
 10         ORG  60000    ;Elegimos esta posicion aunque la
 20         ENT  60000    ;rutina es reubicable
 30         LD   HL,16384 ;Inicio de la memoria de pantalla
 40         LD   B,192    ;La pantalla tiene 192 lineas
 50 BUCLE1  PUSH BC       ;Comienzo del BUCLE1
 60         AND  A        ;Pone a 0 el carry flag
 70         LD   B,32     ;Cada linea tiene 32 bytes
 80 BUCLE2  RRC  (HL)     ;Rotacion circular a la derecha de (HL)
 90         INC  HL       ;Nos ponemos en el siguiente byte
100         DJNZ BUCLE2   ;Cuando se acabe la linea pasamos a 110
110         POP  BC       ;Sacamos el valor introducido en 50
120         DJNZ BUCLE1   ;Si B<>0 hacemos B=B-1 y vamos a 50
130         RET           ;Si B=0 se retorna al BASIC
```

RUTINA NUMERO 1: GIRO DE CARACTERES EN SU POSICION

En esta rutina se gira el contenido de cada uno de los 32 bytes de las 192 líneas de la pantalla (24 filas con 8 líneas por fila), empezando en la posición de memoria 16384 que es el comienzo de la memoria de pantalla.

El cargador BASIC de la rutina podría ser:

```
10 CLEAR 59999: FOR N=60000
   TO 60017: READ A: POKE
   N,A: NEXT N
20 DATA 33,0,64,6,192,197,
   167,6,32,203,30,35,16,251
   ,193,16,244,201
```

Ejecuta la rutina con RANDOMIZE USR 60000

Posiblemente una vez ejecutada la rutina nos demos cuenta de una serie de detalles:

- Esta rutina afecta a la memoria de pantalla, no a la de atributos. Por tanto los colores no sufren cambio ni desplazamiento.

- Si ejecutamos 8 veces la rutina, la pantalla queda en su posición inicial.

- Si en la rutina cambiamos la línea 80 por: 80 BUCLE2 SRL (HL) los caracteres van desapareciendo, y cuando se haya ejecutado 8 veces, habrán desaparecido del todo. En nuestra lista de DATAs deberemos sustituir el 14 que aparece en el décimoprimer lugar por un 62.

- Para que el giro se produzca a la izquierda tenemos que modificar las líneas (Ver listado Rutina 1B):

La lista de DATAs con las modificaciones quedará así:

```
20 DATA 33,255,87,6,192,197
   ,167,6,32,203,22,43,16,
   251,193,16,244,201
```

- Si además queremos que el giro sea a la izquierda pero haciendo desaparecer los caracteres, deberemos sustituir en el listado ya modificado la línea 80 por:

80 BUCLE2 SLA (HL) y en nuestras DATAs de nuevo el número décimoprimero (el 6) por un 38.

RUTINA NUMERO 2: SCROLLING DE PANTALLA PIXEL A PIXEL

Como ya recordarás, en el número 9 de INPUT Sinclair aparecía una rutina de desplazamiento lateral que se había realizado utilizando LDIR, con lo que la pantalla se movía de 8 en 8 *pixel*. Esta vez te ofrecemos una realizada por medio de giros, lo cual permite un desplazamiento mucho más preciso: *pixel* a *pixel*.

Al igual que en la rutina anterior te ofrecemos las claves para que el *scroll* se realice a ambos lados, desapare-

RUTINA 1B

```
 30         LD   HL,22527 ;Deberemos situarnos al final de la me-
                           moria de pantalla.
 80 BUCLE2  RLC  (HL)     ;El giro sera a la izquierda
 90         DEC  HL       ;Vamos retrocediendo desde el final has
                           ta llegar a 16384 (comienzo de pant.)
```

RUTINA 2

```
 10          ORG  60000     ;Situamos aqui el comienzo pero la
 20          ENT  60000     ;rutina es reubicable
 30          LD   HL,16384  ;Inicio de la memoria de pantalla
 40          LD   B,192     ;La$ 192 lineas de una pantalla
 50 BUCLE1   PUSH BC        ;Guarda B y lo saca en 140 (contador)
 60          PUSH HL        ;Mete HL en el stack para pasarlo al
 70          POP  IX        ;registro IX que hace de puntero
 80          LD   A,(IX,+31);Carga A con el ultimo byte de la linea
 90          RRCA           ;Asi consigue pasar al carry el bit 0
100          LD   B,32      ;Hay 32 bytes por cada linea
110 BUCLE2   RR   (HL)      ;'Rotate Right (HL)'
120          INC  HL        ;Nos situamos en el siguiente byte
130          DJNZ BUCLE2    ;Cierra el bucle abierto en 100
140          POP  BC        ;Saca el valor metido en 50
150          DJNZ BUCLE1    ;Si B-1<>0 THEN GOTO 50
160          RET            ;Si B-1=0 Retorna al BASIC
```

ciendo o no, etc. En cualquier caso te recomendamos que compares bien ambos métodos de *scrolling*, ya que, de ello, sacarás mucho provecho para tus conocimientos (Ver listado Rutina 2).

Con esta rutina conseguimos que se desplace la pantalla *pixel* a *pixel* y que a su vez vuelva a aparecer por el lado izquierdo. No olvidemos que estamos tratando con la memoria de pantalla compuesta de puntos o bits (1 ó 0), por lo que para conseguir el efecto de la «vuelta a salir» nos situamos en el byte 32 de cada línea (IX+31). Una vez pasado este byte al A hacemos RRCA con lo que pasamos el bit 0 (el primero que se pierde por la derecha) al acarreo. Más tarde al hacer RR (HL) introducimos el acarreo en la posición del bit 7 del primer byte de la línea. Con ello consegimos pasar el bit 0 del byte más a la derecha, a la posición del bit 7 del byte más a la izquierda. Así no perdemos ni un sólo bit de información. Te recomendamos para una mejor comprensión que consultes los gráficos incluidos al principio en este artículo.

El cargador BASIC de la rutina será:

```
10 CLEAR 59999: FOR N=60000
   TO 60023: READ A: POKE
   N,A: NEXT N
20 DATA 33,0,64,6,192,197,
   229,221,225,221,126,31,
   15,6,32,203,30,35,16,251
   ,193,16,238,201
```

Como sugerencia para que puedas comprobar mejor el funcionaminto, te aconsejamos que una vez cargadas las DATAs o el programa en C.M. hagas:

FOR N=0 TO 255: RANDOMIZE USR 60000: NEXT N, con lo que conseguirás un desplazamiento de la pantalla hasta dejarla en su posición inicial.

Para conseguir la misma rutina pero con el desplazamiento a la izquierda deberemos hacer los cambios que se muestran en la Rutina 2B.

La nueva línea de DATAs será:

```
20 DATA 33,255,87,6,192,197
   ,229,221,225,221,126,225
   ,7,6,32,203,22,43,16,251
   ,193,16,238,201
```

Si queremos realizar un desplazamiento a la derecha pero sin que vuelva a salir, la rutina se acorta bastante (Ver Rutina 3).

La instrucción RR (HL) introduce en el bit 7 el valor contenido por el acarreo, que en un principio ha sido puesto a cero con AND A. Esto ocurre con los bytes situados al comienzo de las 192 líneas, por lo que el resultado es que la pantalla irá desapareciendo a la vez que se desplaza.

El cargador BASIC será:

RUTINA 2B

```
 30          LD   HL,22527 ;Nos situamos al final de la memoria de
 80          LD   A,(IX-31);pantalla y vamos retrocediendo con DEC
 90          RLCA          ;hasta llegar al principio (16384). Los
110 BUCLE2   RL   (HL)     ;giros como es logico seran a la
120          DEC HL        ;izquierda.El razonamiento sera similar
                           al hecho de antes. Las demas lineas se
                           quedaran igual.
```

RUTINA 3

```
 10          ORG  60000    ;Como podras comprobar esta rutina es
 20          ENT  60000    ;muy parecida a la numero 1. El cambio
 30          LD   HL,16384 ;radica en la instruccion de la linea
 40          LD   B,192    ;80. Es un ejemplo donde puedes
 50 BUCLE1   PUSH BC       ;comprobar que utilizando instrucciones
 60          AND  A        ;semejantes del mismo grupo se pueden
 70          LD   B,32     ;conseguir resultados totalmente
 80 BUCLE2   RR   (HL)     ;diferentes,en nuestro caso un
 90          INC  HL       ;desplazamiento de la pantalla completa
100          DJNZ BUCLE2   ;o un giro de cada caracter en su
110          POP  BC       ;posicion. Una vez mas te recomendamos
120          DJNZ BUCLE1   ;que consultes los graficos incluidos
130          RET           ;al principio.
```

```
10 CLEAR 59999: FOR N=60000
   TO 60017: READ A: POKE
   N,A: NEXT N
20 DATA 33,0,64,6,192,197,
   167,6,32,203,14,35,16,251
   ,193,16,244,201
```

Para conseguir el desplazamiento al lado contrario hacemos los cambios habituales (Ver Rutina 3B).

La lista de DATAs quedará:

```
20 DATA 33,255,87,6,192,197,
   167,6,32,203,6,43,16,251,
   193,16,244,201
```

En esta ocasión hemos presentado dos rutinas que se dividen en sus 8 variantes y con las que esperamos que haya quedado claro el tema de los giros y desplazamientos, pero el C.M. ofrece más posibilidades para el manejo de la pantalla. Esto y mucho más será el contenido de futuros artículos.

RUTINA 3B

```
 30          LD   HL,22527
 80 BUCLE2   RL   (HL)
 90          DEC  HL
```

BUSCA TU NOMBRE

UN CICLOMOTOR VESPINO

* ANTONIO FERNANDEZ DELGADO, ALCALA DE HENARES (MADRID).

UNA CADENA HIFI DE SPECTRAVIDEO

** JUAN LOPEZ SALA, ALICANTE.
** EMILIO DE ANDRES DE LA VEGA, MADRID.
* J. LORENZO LAGO SOLANA, R. DE CAMARGO (CANTABRIA).

UN LP DE GABINETE CALIGARI

** SANTIAGO CAMPAYO CUESTA, BADALONA (BARCELONA).
** ALBERTO VILARMAU ESPINAL (BARCELONA).
** LUIS ALBERTO CARRION MORENO, VALLADOLID.
** MANUEL CASTILLEJOS VAYA, MADRID.
** MANUEL ALONSO MARTINEZ, BILBAO.
** PEDRO BARAHONA CORDERO, COSLADA (MADRID).
** MIGUEL GUNE RODRIGUEZ, BARCELONA.
** JOSEBE FERNANDEZ BAREA, SANTURCE (VIZCAYA).
** JOSE LUIS MARTINEZ MUNOZ, STA. COLOMA DE GRAMANET (BARCELONA).
* GONZALO NOVOA SOTELO, ORENSE.

UN LOTE DE SOFTWARE DE ERBE

** SANTIAGO ESTEFANIA BARRIO, GETXO (VIZCAYA).
** J. MANUEL COLOMINA MESTRES, STA. COLOMA DE GRAMANET (BARCELONA).
** VICTORINO RODRIGUEZ VALDESOGO, LEON.
** J. MANUEL PEREIRA GARCIA, BARCELONA.
** MA. ANGUSTIAS DELGADO, GRANADA.
** MANUEL TERRER MENA, CARTAGENA (MURCIA).
** MIGUEL SALAS GARCIA, MADRID.
** J. CARLOS DE LA LLANA SANTO, REUS (TARRAGONA).
** SERGIO MARTINEZ COLLDEFORN, BARCELONA.
* ROBERTO TORRIJOS SANTOS, TALAVERA (TOLEDO).

TRES PROGRAMAS DE COMPULOGICAL

** FERNANDO GONZALEZ RODRIGUEZ, LA OROTAVA (TENERIFE).
** J. ANTONIO PONZO GALLETERO, CHIVA (VALENCIA).
** A. MANUEL YANEZ VALEIRO, BILBAO.
** JORDI MARZO VIDAL, BARCELONA.
** JOAN CARRASCO DOMINGUEZ, TERRASA (BARCELONA).
** ANGELMO FERNANDEZ BERRODAS, MALAGA.
** FRANCISCO MACIA LACUSA, BARCELONA.
** ANDRES ORTIZ MENDEZ, MARTOREL (BARCELONA).
** ESTEBAN HORCAJO ARROYO, S.S. DE LOS REYES (MADRID).
* JAVIER SUBIRATS CERVERA, REUS (TARRAGONA).

TRES PROGRAMAS DE PROEINSA

** JORDI PRUNES CASTERAS, TERRASA (BARCELONA).
** DAVID BRAVO CALVO, BARCELONA.
** ANTONIO GRIMALT ANDREU, MANACOR (MALLORCA).
** RUBEN DOMINGUEZ LOPEZ, AGUADULCE (ALMERIA).
** SALVADOR DE LA FUENTE FERNANDEZ, VALENCIA.
** ALFONSO MANTECON LAGE, LAS ROZAS (MADRID).
** MIGUEL LA TORRE ROVIROSA, VILAFRANCA DEL PENEDES (BARCELONA).
** M. ANGEL FLORES ACUNA, LA CORUNA.
** J. CARLOS GAITAN ALFARO, CORDOBA.
* J. MIGUEL TORES CARO, CADIZ.

TRES PROGRAMAS DE MICROUNO

** MANUEL PINEIRO DESADA, PORRINO (PONTEVEDRA).
** A. JOSE PALACIOS SOLER, VALENCIA.
** SEBASTIAN MONZON PEREZ, SEVILLA.
** J. CARLOS CAFAFONS CUBERO, SITGES (BARCELONA).
** JAVIER HINOJOSA RUIZ, CASTELLAR DEL VALLES (BARCELONA).
** ALFREDO CANTELI SUAREZ, OVIEDO (ASTURIAS).
** R. ALBERTO GARGALLO CASCAN, BARCELONA.
** SANTIAGO FERREIRO PARAMO, LA CORUNA.
** ANTONIO GALAN ELEZ, MADRID.
* JESUS VALERO ALCARAZ, MADRID.

CUATRO PROGRAMAS DE SERMA

** ANTONIO GOMEZ AGUILAR, MALAGA.
** PAQUI RODRIGUEZ GONZALEZ, BADAJOZ.
** FERNANDO ROJAS PALACIOS, ALGECIRAS (CADIZ).
** RAFAEL ASENJO PLAZA, MARACENA (GRANADA).
** J. MANUEL PARRA BARRERO, LA HERRADURA (GRANADA).
** PABLO HERRERA PEREZ GRUESO, MORA (TOLEDO).
** TEOFILO DEL RINCON GONZALEZ, PUIG DEN VALS (IBIZA).
** JOSE LUIS GONZALEZ OLEA, FUENLABRADA (MADRID).
* JAIME SOLA FLOX, VILANOVA I LA GELTRUS (BARCELONA).
* ALVARO DAPENA URCOLA, ZARAGOZA.

UN KIT ROBOTICO DE FERRE-MORET

* JOSE JULIO VELAZQUEZ, LEJONA (VIZCAYA).

UN JOYSTICK DE COMPULOGICAL

** MIGUEL MILLAN DE LA TORRE, S.JUAN DESPI (BARCELONA).
** VICENTE MORALEDA HIDALGO, SEVILLA.
* RAMON JUNQUERA SOBRON, GALDAKANO (VIZCAYA).

UN JOYSTICK DE MICROUNO

** JOSE C. RODRIGUEZ ESCOBAR, MADRID.
** BENITO SANTAMARIA ROMERO, MADRID.
** JOSE ANTONIO GONZALEZ CUETO, GIJON (ASTURIAS).
** DANIEL ROBS ESTRAGUEZ, BARCELONA.
** ANTONIO DIAZ GONZALEZ, SANTANDER.
** EDUARDO TORRECILLAS VALVERDE, BADALONA (BARCELONA).
** PEDRO LANCILLA TELLA, ZARAGOZA.
** V. ANTONIO UCEDA ALVAREZ, GETAFE (MADRID).
** J. MANUEL VIGO COUSINO, VIGO (PONTEVEDRA).
** MARIANO LOPEZ LOPEZ, COLMENAR VIEJO (MADRID).
** JAVIER CORONA DELGADO, PAMPLONA.
** FCO. JAVIER DIAZ GARZON, BARCELONA.
** J. MANUEL TRIAS COMPANY, SOLLER (MALLORCA).
** PABLO VAQUERO SEPULVEDA, AVILA.
* JESUS OCANA GOMEZ, MADRID.

UN CRONOMETRO DE MICROUNO

** JUAN PENA SANCHEZ, ISLA CRISTINA (HUELVA).
** ANGEL GOMEZ LOPEZ, LUGO.
** IGNACIO TIERNO MARTINEZ, BARCELONA.
** MA. LUISA SANZ AGUADO, LLAVERNERS (BARCELONA).
* J. LUIS BLASCO CABAL, MADRID.

UN LIBRO DE ANAYA MULTIMEDIA

** ANTONIO CABEZA HERNANDEZ, TALAVERA (TOLEDO).
** JAVIER MARTINEZ CORMENZANA, BILBAO.
** MIGUEL SERRANO DEL MANZO, CACERES.
** ECKART SCHILDER, BREXO (LA CORUNA).
** SANTIAGO DIAZ PEREZ, PORTUGALETE (VIZCAYA).
** RICARDO SEMPERE PALAU, QUART DE POBLET (VALENCIA).
** FRANCISCO TINEO RAMIREZ, MATARO (BARCELONA).
** MIGUEL ANGEL FERNANDEZ, MADRID.
** JORGE ABAD GIL, MADRID.
** TEJVA GOMEZ RODRIGO, SALAMANCA.
** J. JAVIER VAZQUEZ SERRANO, CORDOBA.
** J. CARLOS GARRIDO BERNARDEZ, ORENSE.
** J. MIGUEL MELLADO BREY, SAN ROQUE (CADIZ).
** SANTIAGO HERNANDEZ SANCHEZ, FUENLABRADA (MADRID).
** GINES CLIMENT PEREZ, VILLAJOYOSA (ALICANTE).

LOS MEJORES DE INPUT SINCLAIR

PUESTO	TITULO	PORCENTAJE
1.°	Commando	17,5 %
2.°	Green Beret	15,7 %
3.°	Movie	15,1 %
4.°	Batman	9,1 %
5.°	Saboteur	8,4 %
6.°	The way of the tiger	7,4 %
7.°	Phantomas II	7,3 %
8.°	Skyfox	7,2 %
9.°	Rambo	6,2 %
10.°	Sir Fred	6,1 %
		100 %

Para la confección de esta relación únicamente se han tenido en cuenta las votaciones enviadas por nuestros lectores de acuerdo con la sección «Los Mejores de Input».

Octubre de 1986.

XARQ

Xarq es un programa de muy reciente aparición, producido por **Electric Dreams**, que nos ha sorprendido gratamente por la calidad de sus gráficos y por la espectacularidad de sus pantallas. Se trata de uno de esos juegos en los que la atención, la práctica y los

DATOS GENERALES

TITULO Xarq

FABRICANTE Electric Dreams

ORDENADOR Spectrum 48

TEMA DEL PROGRAMA
Arcade

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	8
INTERES	7
GRAFICOS	7
COLOR	8
SONIDO	5
TOTAL	35

el atractivo de su tratamiento gráfico, especialmente el color, y la velocidad de los desplazamientos, contribuye a crear un ambiente emocionante. En suma, un buen programa.
En cuanto al grado de dificultad, debemos decir que es bastante alto. Es posible que los más novatos en estas lides se desanimen un poco al principio, pero la verdad es que no se pierde el interés por grandes que

reflejos juegan un papel primordial, sin carecer por ello de una cierta dosis de estrategia.
El argumento es bastante simple, como corresponde a esta clase de programas, pero con la originalidad y el interés suficientes como para que no resulte aburrido. Por otra parte, sean los descalabros.
Terminamos recomendando este programa a los entusiastas de los juegos «de mucha acción».

★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★

ACE: MISION DE COMBATE

Aquellos que ya hacían sus primeros «pinitos» con un 16K allá por 1982, recordarán que entre las tediosas aventuras de un monigote llamado **Horacio**, los plagios mal hechos del **Comecocos**, y los programas de ajedrez sin «enroque» ni «peón al paso», destacaba un formidable simulador de vuelo que todos codiciaban y muy pocos poseían, a pesar del incipiente «pirateo».

DATOS GENERALES

TITULO ACE

FABRICANTE Cascade Games

ORDENADOR Spectrum 48

TEMA DEL PROGRAMA
Simulador de vuelo

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	8
INTERES	9
GRAFICOS	9
COLOR	7
SONIDO	6
TOTAL	39

Tampoco habrán olvidado dos magníficos programas, también simuladores que aparecieron poco después, adelantados en un par de años a su tiempo, con los nombres de **Fighter Pilot** y **Combat Lynx**.
Pues bien, hemos tenido que esperar desde entonces varios años a que **Cascade Games** lanzara un programa simulador de semejantes características, que superase a los

anteriores: «**ACE**».
Técnicamente, **ACE** no reproduce las condiciones reales de vuelo con tanta fidelidad como otros simuladores más complejos (como el propio **Fighter Pilot**, por ejemplo), pero esto, lejos de ser un inconveniente, contribuye a hacer más asequible el programa a los «novatos», y a aumentar la rapidez y el interés.
Con **ACE** es posible hacer acrobacias sin sufrir descalabros ni perder el control del aparato, y combatir sin estar más pendiente de mantener el avión en vuelo que de derribar al enemigo. Sus creadores se han esforzado por reproducir el ambiente y la emoción del combate aéreo, y por mejorar la tradicionalmente pobre estética de este tipo de programas, simplificando el manejo de los mandos, introduciendo distintos tipos de objetivos en tierra, mar y aire, y creando un paisaje.
En cuanto al argumento del programa, podemos resumirlo así:
El enemigo ha desembarcado gran cantidad de tropas en tus costas, sin encontrar ningún tipo de resistencia. A la vanguardia, avanzan poderosas formaciones de carros de combate protegidos por misiles tierra-aire «SAM» y numerosos helicópteros, además de varias escuadrillas de mortíferos cazas de combate. Las únicas fuerzas disponibles para hacer frente al enemigo y detener la invasión, se reducen a tres aviones AWAT y un sólo piloto: tú, naturalmente.
La misión consiste en derribar todos los cazas enemigos, destruir las tropas terrestres invasoras, y atacar la flota mientras se realiza la evacuación subsiguiente a la derrota.
Probablemente, necesitarás semanas de práctica para conseguirlo.
Lo que más nos ha gustado de **ACE**, sin lugar a dudas, es el tratamiento dado al combate aéreo, tanto por su impecable resolución gráfica como por el emocionante efecto de realismo.
El único aspecto negativo que, a nuestro juicio, presenta el programa, es el sistema de protección empleado. Se trata de algo nuevo y original pero a la vez molesto y engorroso. Antes de acceder al programa, aparece en la pantalla un jeroglífico indescifrable que, mirado a través de una lente especial a la distancia adecuada, se convierte en un código legible de letras y números, que hay que introducir por el teclado. El problema es que dicho

código aparece siempre borroso y confuso, y además, es necesario graduar previamente su formato según las dimensiones de la pantalla.
Necesitarás un buen rato para comprender el funcionamiento del artilugio de marras (y no por comprenderlo deja de dar problemas), y para colmo, las instrucciones de manejo están en inglés.
Terminaremos diciendo que **ACE** es un excelente programa, digno de figurar como «lo mejor» que se ha hecho hasta ahora en materia de simuladores de vuelo. Lo recomendamos especialmente no sólo a los aficionados a este tipo de programas, sino a todos los amantes de la emoción y la acción trepidante.

★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★

SECUESTRO!!

Después de ver tantas pequeñas «obras maestras» como han ido apareciendo en los últimos meses (precisamente comentamos una de ellas en este número, aparte de la que ahora nos ocupa), creímos que ya habíamos perdido nuestra capacidad de asombro. Pues bien, la verdad es que no nos ha costado mucho encontrar otro programa capaz de dejarnos «con la boca abierta»: **Hijack**, una de las últimas creaciones de **Electric Dreams**.
Lo primero que hemos de destacar de este magnifico programa, es su originalidad. Tanto el tema como su tratamiento son absolutamente novedosos, hasta el punto de que casi podríamos decir que se ha inaugurado un nuevo género. El juego presenta una gama de posibilidades excesivamente amplia como para describirla aquí con detalle, así que vamos a tener que limitarnos a hacer un breve resumen del argumento:
Acabas de obtener un cargo de responsabilidad en el Departamento de Secuestros del gobierno. Tu misión consiste en negociar con un grupo de secuestradores antes de que expire el plazo límite, obteniendo previamente toda la información posible, y el apoyo político y financiero necesario para realizar la operación con éxito y salvar a los rehenes. Lo primero que debes hacer, es familiarizarte con los diversos departamentos del edificio sede de las oficinas, conocer los medios para sacar el máximo partido a los empleados a tu servicio, y mantenerte en buenas relaciones con aquellos compañeros que puedan serte útiles, con la prensa, y sobre

todo, con el presidente. No olvides tener un ojo puesto en el Washington Post, y otro en los empleados envidiosos que pretendan hundirte. Tampoco descuides los recursos económicos a tu disposición, pues negociar con terroristas suele resultar bastante caro. Una vez que hayas obtenido la información y el apoyo necesarios, deberás dirigirte al presidente, y solicitarle permiso para intentar negociar, desplazándote al lugar del conflicto en helicóptero. Si tienes mala prensa, y el presidente no está satisfecho con tus gestiones, su respuesta será negativa.
Finalmente, si consigues ponerte en contacto con los secuestradores, tendrás que hacerles una buena oferta para disuadirles de su empeño. Una vez logrado esto, habrás obtenido prestigio, medallas, y la confianza de tus superiores; si no, serás despedido.
Debemos advertir, para terminar, que el programa es de una gran complejidad, y requiere un alto grado de concentración. Sus autores han conseguido obtener una síntesis genial entre los juegos simuladores de gestión, y los de aventuras, apoyándose en un argumento original y un tratamiento del tema que permite un sinfín de posibilidades. Los amantes de los juegos de estrategia bien hechos, con éste van a disfrutar como nunca.

DATOS GENERALES

TITULO Hijack

FABRICANTE Electric Dreams

ORDENADOR Spectrum 48

TEMA DEL PROGRAMA
Estrategia

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	10
INTERES	9
GRAFICOS	7
COLOR	7
SONIDO	7
TOTAL	40

★★★★★★★★★

DYNAMITE DAN, SEGUNDA PARTE

El doctor **Blitzen** ataca de nuevo!! Esta vez, ciego de rabia por su última derrota a manos del agente **Dan**, decide esclavizar a todos los jóvenes del mundo, propagando a través de los discos unas ondas sonoras que anulan la voluntad. Tu misión, naturalmente, consiste en impedirlo.
La aventura se desarrolla en ocho islas. Debes ir de una en una buscando los discos, y pinchándolos en el tocadiscos correspondiente. En cada isla, existe un sólo disco y un sólo tocadiscos. Para pasar de una isla a otra, puedes hacerlo montado en tu Zeppelin, cargándolo previamente de combustible. Una vez llegado a la octava isla, deberás colocar una bomba en el laboratorio del **Dr. Blitzen** y escapar, pero ¡ojo!, porque si no llevas puestas unas gafas especiales, no podrás resistir sus poderes hipnóticos.
Las bombas te serán muy útiles para abrirte paso cuando encuentres una puerta cerrada, y la comida será imprescindible para que puedas seguir adelante. Por otra parte, también existen otros objetos con diversos usos: algunos te proporcionarán poderes especiales, como por ejemplo la invisibilidad, y otros te permitirán acceder a los pasadizos secretos que poseen todas las islas.
Los gráficos y las soluciones técnicas del juego son prácticamente los

DATOS GENERALES

TITULO Dynamite Dan II

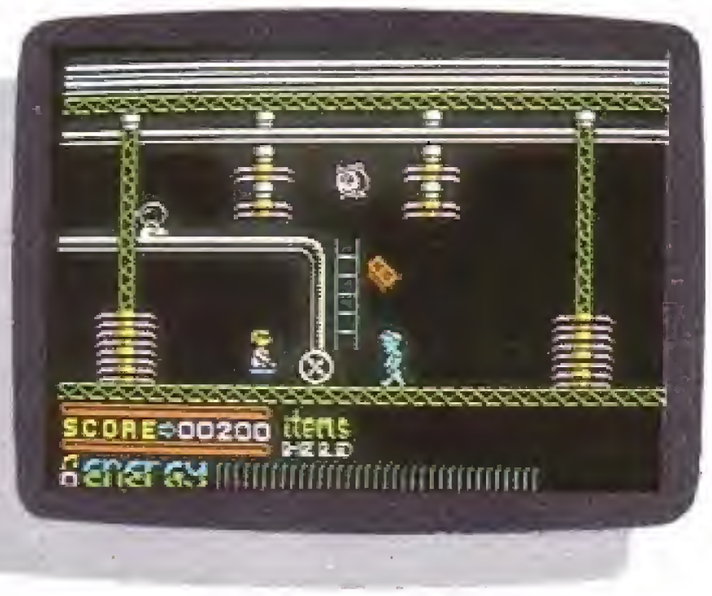
FABRICANTE Mirrorsoft

ORDENADOR Spectrum 48K

TEMA DEL PROGRAMA
Arcade

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	5
INTERES	7
GRAFICOS	7
COLOR	7
SONIDO	6
TOTAL	32



mismos que vimos en la primera parte, aunque se han añadido algunas mejoras. El argumento y el planteamiento general son más interesantes, y la gran variedad de objetos que se han introducido contribuyen a aumentar las posibilidades de este divertido programa, con respecto a la versión anterior. No obstante, advertimos a aquellos que ya hayan jugado con el **Dynamite Dan I** que no esperen encontrar nada nuevo.
Aunque no se ha mejorado el programa tanto como nos hubiera gustado, la verdad es que no deja de ser bastante bueno. Al menos, tan bueno como su primera parte.

★★★★★★★★★

JACK, THE NIPPER

Ha nacido una estrella: el travieso y recalcitrante bebé «**Jack, el mico**». Seguro que siempre has deseado hacer alguna gamberrada tan gorda que no te has atrevido. Pues bien, este el el momento de hacer no una, sino todas las que puedas, pues en eso precisamente consiste el juego.
Existen trece tipos distintos de cosas útiles para crear el caos a tu alrededor, aparte de una mortífera cerbatana, ideal para incordiar a los mayores.
El objetivo consiste en recoger todos los objetos y exprimir tu imaginación hasta dar con su uso adecuado. Si eres un buen gamberro, no te costará mucho.
Los escenarios son tridimensionales, y la acción transcurre en el interior de la casa de **Jack**, en las calles y en los comercios de su barrio.
No estaría bien que te reveláramos todos los secretos del juego, pero si te vamos a hacer algunas sugerencias:
Nada más levantar a **Jack** de la cama, al comenzar el juego debes coger la cerbatana subiendo al estante donde se encuentran la pelota y el osito de peluche. Con ella, podrás disparar a tu padre, tu madre, tu hermanita, el perro, a las abuelas, los guardias, los dependientes, etc, etc, obteniendo con ellos algunos (pocos) puntos.
Después, puedes divertirte un rato cortocircuitando la tienda de ordenadores con la pila, o parando la fábrica de calcetines dejando caer una pesa en la cadena de confección.

DATOS GENERALES

TITULO Jack, The Nipper

FABRICANTE Gremlin Graphics

ORDENADOR Spectrum 48

TEMA DEL PROGRAMA
Travesuras

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	7
INTERES	8
GRAFICOS	9
COLOR	8
SONIDO	7
TOTAL	39

¿Qué se puede hacer con el tubo de pegamento? ¿Qué mejor que pegarle las dentaduras postizas al protésico?

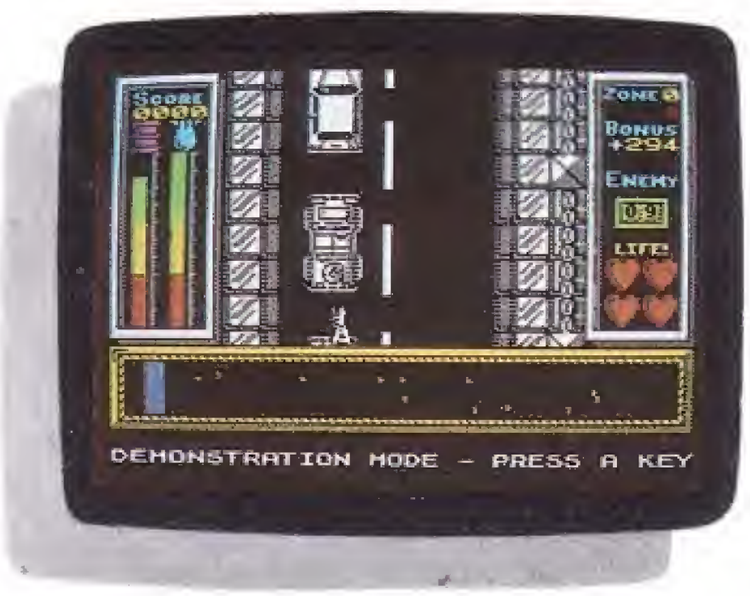
Sintiéndolo mucho, no podemos decirte más. Pero estamos seguros de que tu retorcida mente no tendrá muchos problemas.

Jack, The Nipper, es sin duda el programa más simpático y entretenido que hemos tenido oportunidad de probar.

★★★★★★★★★★

STAINLESS STEEL

Leyendo los reclamos de la carátula, nos encontramos con una inmodesta afirmación que transcribimos literalmente: «El juego tiene el *scroll* bidireccional más rápido y suave jamás visto en un ordenador».
Después de hacer las comprobaciones pertinentes, no sin cierto escepticismo, nos vimos obligados a «quitarnos el sombrero» y a reconocer que, efectivamente,



era así. No obstante, y en honor a la verdad, cambiaríamos la última palabra, «ordenador», por **Spectrum**.

DATOS GENERALES

TITULO Stainless Steel

FABRICANTE Mikro-Gen

ORDENADOR Spectrum 48-128K

TEMA DEL PROGRAMA
Coche fantástico

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	8
INTERES	8
GRAFICOS	9
COLOR	6
SONIDO	6
TOTAL	37

Naturalmente, hay otros programas para otros ordenadores con un *scroll* más rápido, más suave, y más todo lo que se quiera.
Dicho esto, y calificando con un soberano «nueve» a los gráficos del programa, pasaremos a comentar brevemente su argumento.
El juego se divide en cuatro zonas, con escenarios diferentes, creciente grado de dificultad, y distintos tipos de acción. En la zona cero, hay que combatir a pie, hasta llegar al coche «Viento Nocturno», en el que transcurre el resto de la aventura.

En las zonas uno y dos, el objetivo es destruir a todos los enemigos que aparezcan en el camino, para poder acceder a la zona tres, donde se encuentra el malvado **Dr. Vardos**.
Derrotando a este perverso personaje, se habrá culminado la misión con éxito.
Para facilitar un poco las cosas, en la parte inferior de la pantalla hay un radar «buscaenemigos». El inconveniente que tiene es que todo ocurre tan deprisa, que si se aparta

un momento la vista para consultarlo, se corre el riesgo de estrellarse.
En definitiva, **Stainless Steel** es un programa de una excelente calidad gráfica, con un grado de interés muy alto. El único punto negativo que hemos encontrado, es el hecho de que sus creadores hayan tenido que sacrificar el color para conseguir un *scroll* tan impresionante. A pesar de ello, mandamos desde aquí nuestra más efusiva felicitación a los señores de **Mikro-Gen**.

★★★★★★★★★★

PARTIDO DE TENNIS

Las firmas **Imagine** y **Konami** en colaboración, han lanzado al mercado recientemente un nuevo simulador de **Tennis**, que aporta una gran variedad de mejoras con respecto a los pocos programas de este tipo existentes hasta el momento.
El juego se desarrolla en un escenario tridimensional, con fiugras de considerable tamaño, y una resolución gráfica muy clara. Los

partidos pueden jugarse con uno o dos jugadores (en modalidad de un jugador, el ordenador controla al contrincante), o de «**dobles**», con cuatro tenistas en la pantalla.
El manejo del programa es muy sencillo, sin que por ello se desvirtúe su efecto de realismo. Con sólo cuatro teclas, además del botón de disparo, es posible controlar los desplazamientos y los golpes de los jugadores, teniendo en cuenta los ángulos de trayectoria de la pelota, y la puntería en los saques. También es posible desarrollar todas las tácticas tradicionales en el tenis, como «subir a red», golpear de «volea», ensayar saques largos, o hacer una elegane «dejada», amén de muchas otras interesantes posibilidades.
Los gráficos no son especialmente

DATOS GENERALES

TITULO Tennis

FABRICANTE Imagine-Konami

ORDENADOR Spectrum 48

TEMA DEL PROGRAMA
Deportivo

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	7
INTERES	8
GRAFICOS	7
COLOR	6
SONIDO	5
TOTAL	33

brillantes, y el color tampoco contribuye a dar vistosidad al juego, pero teniendo en cuenta las peculiares características de esta clase de programas, no debemos dar excesiva importancia a este detalle.
Por lo demás, el juego es casi perfecto. Aprovechamos la ocasión para recomendarlo especialmente a los amantes de este deporte.

★★★★★★★★★

EL COCHE FANTASTICO

Hace varios meses, **Ocean** nos sorprendió con un magnífico programa basado en la popular serie de televisión «**V**», del que publicamos, como recordaréis, un mapa y un detallado comentario.
Pues bien, en esta ocasión volvemos a encontrarnos con un nuevo programa de la misma firma, apoyado en otro éxito televisivo: **Knight Rider** (El Coche Fantástico).
El juego consiste en recorrer diversas ciudades de los Estados Unidos, abortando los planes de un grupo de terroristas. Para ello, es preciso seguir ls instrucciones que se reciben a través del ordenador **Kitt**, y cumplir algún tipo de misión en distintas partes del país.
Los escenarios son básicamente de dos tipos, que se corresponden con las dos partes de que consta cada objetivo:
- El desplazamiento por carretera desde una ciudad a otra, a bordo del **Coche Fantástico**.
- El desarrollo final de cada misión, con **Michael Knight** actuando fuera de su vehículo, en el interior de una de las bases de los terroristas.

Al igual que ocurre en la serie de televisión, el protagonista nunca lleva armas, y debe lograr sus

DATOS GENERALES

TITULO Knight Rider

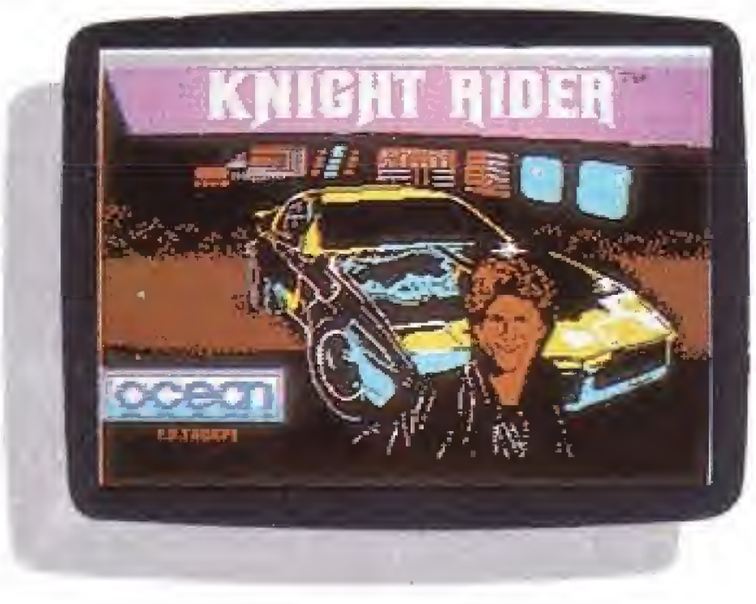
FABRICANTE Ocean

ORDENADOR Spectrum 48

TEMA DEL PROGRAMA
Coche fantástico

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	7
INTERES	6
GRAFICOS	7
COLOR	6
SONIDO	6
TOTAL	32



objetivos usando sólo su astucia. No obstante, el Coche Fantástico sí va armado con un potente láser para combatir a los helicópteros enemigos que interfieran sus desplazamientos.
Haciendo un balance general del programa, hemos de decir que presenta ciertos aspectos que no nos han gustado, sobre todo si lo comparamos con «V». El repertorio gráfico no alcanza la calidad a que

Ocean nos tiene acostumbrados, y el grado de interés no es lo suficientemente alto como para merecer una buena nota. Pensamos que se podia haber sacado mucho más partido al tema.
Naturalmente, esto no quiere decir que no estemos ante un buen programa, pero si que puede decepcionar un poco a aquellos que esperen demasiado de él.

★★★★★★★★★

RESCATE EN FRACTALES

Derrotados una vez tras otra en el espacio abierto, los temibles **Jaggies** se han refugiado en uno de los planetas más inhóspitos de toda la galaxia: **Fractalus**.
Tu misión consiste en atravesar las líneas de defensa y rescatarlos, localizando los restos de sus naves y aterrizando en algún lugar próximo.
Inicialmente, comienzas despegando (de forma automática, es decir, sin

DATOS GENERALES

TITULO Rescue on Fractalus

FABRICANTE Activision

ORDENADOR Spectrum 48

TEMA DEL PROGRAMA
Rescate Espacial

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	8
INTERES	5
GRAFICOS	6
COLOR	6
SONIDO	5
TOTAL	30

los gráficos del paisaje por el que se supone, se desplaza la nave.
El paisaje es fractal y el ordenador lo va generando según la nave se mueve a un lado o a otro, de fomra que éste va cambiando continuamente.
Sin embargo, pese a lo espectacular del procedimiento, esto conlleva la limitación de la velocidad; el movimiento es lento.

que tu mismo puedas controlar el aparato) de una nave nodriza. adentrándote en el espacio y penetrando en la atmósfera del planeta.
El aterrizaje también se realiza sin que medie tu intervención. Simplemente, basta con pulsar la tecla adecuada en el momento preciso.
Lo único reseñable de este programa está en la forma en que se elaboran

Además el juego tiene serias limitaciones. No es posible determinar la distancia a los accidentes del paisaje y, ni siquiera, se puede colisionar contra ellos.

★★★★★★★★★

EL INCREIBLE BOMBERO

Un desgraciado accidente ha provocado una reducción en el tamaño de nuestro protagonista, que a partir de este momento inicia la búsqueda de la solución a su problema.
Para retornar a su tamaño original precisa encontrar cinco claves que están esparcidas por una gran casa llena de habitaciones, salas y peligros.
La tarea no es nada fácil y se

DATOS GENERALES

TITULO Incredible Shirinking Fireman

FABRICANTE Mastertronic

ORDENADOR Spectrum 48K

TEMA DEL PROGRAMA
Bombero en apuros

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	6
INTERES	6
GRAFICOS	8
COLOR	7
SONIDO	6
TOTAL	33

complica con la existencia de seres extraños que tratan de estropear la misión, consumiendo, al rozarle, parte de su energía.
El acceso a la habitaciones no es siempre posible y aunque en la parte superior de la pantalla unas flechas indican si se puede continuar en esa dirección, en algunos casos no se permite el acceso. La razón es que a determinadas habitaciones sólo se puede llegar si se transporta un objeto determinado, por lo que una parte de misión del jugador es conocer qué se necesita para atravesar estas salas.
El protagonista puede moverse a izquierda y derecha así como saltar, disponiendo de un total de cinco vidas para conseguir su objetivo.
Los objetos no están identificados en pantalla donde sólo se aprecia una especie de diamante que los define, por lo que es necesario cogerlos para saber de que se trata. Podemos encontrarnos con llaves, relojes digitales, tarjetas, y hasta ratones, pero sólo es posible transportar cinco objetos simultáneamente, por lo que en determinados momentos es necesario dejar alguno de los que llevamos encima.
No todos los objetos son de utilidad por lo que no es preciso transportarlos, pero sólo la práctica nos indicará cuáles.
Otra peculiaridad del juego es que los objetos no varían de situación lo cual facilita localizarlos.
Un juego de **Mastertronic** con buenos gráficos y color en el que el pequeño bombero debe volver a su tamaño natural.

EL ZOCO DE INPUT

Todo se compra y se vende. Los antiguos zocos fueron lugares destinados a todo tipo de transacciones. INPUT también tiene el suyo. Vuestras operaciones de compra, cambio o venta serán publicadas en esta sección, pero dos son las limitaciones que imponemos:

a) La propuesta tendrá que ver con la microinformática.
b) Nos reservamos el derecho de no publicar aquellos insertos de los que se sospeche un trasfondo lucrativo.

Ahora un ruego. Tratar de resumir al máximo el texto; escribir casi como un telegrama siendo claros y concisos.

Envía tu mensaje a:

INPUT SINCLAIR-ZOCO
P.º de la Castellana, 93. Planta 14
28046 **MADRID**

EL ZOCO

Cambio novedades de Spectrum tengo novedades como Nodes of yesod, West Bank, Supertest, series Basketball, etc. Interesados dirigirse a:

Miguel Angel Garcia Navarro
Avda. San Fernando, 31
Peñaflor (Sevilla)

Cambio cinta de juegos por joystick o interface multijoystick. También cambio juegos (Zorro, Surizam, Winter Games/Sports, etc).

Jesús
Tel. (93) 890 42 04

Intercambio programas para ZX-Spectrum. Tengo siempre lo último. Escribid a:

Carlos Echevarria
Maiatzren Bata, 2, 4ºB
Tel. (94) 464 31 94
Lejona (Vizcaya)

Compro los siguientes programas para el ZX Spectrum: Astrología y Betabasic de la casa Ventamatic o los cambio por otras utilidades o juegos. Enviar ofertas a

Amador Ramírez Ibanez
Alhambra, 3
Albolote (Granada)

Vendo video juego Atari 2600, 5 cartuchos de juego, 2 mandos (joystick). Precio a convenir.

Marco Cruz Olías
General Franco, 20
Tel. 53 10 19
Yuncler (Toledo)

Desearia contactar con usuarios del Spectrum de todas partes de España para intercambio de juegos, programas e ideas.

Victor Mexia
Gaztambide, 21, 2ºB
Tel. (91) 449 70 91
28015 Madrid

¿Queréis aprender alemán o inglés con vuestro ordenador? Yo tengo un curso que os puede interesar. Escribidme y os enviaré información totalmente gratis.

Carmen Abella
Avda. Portugal, 32, 4ºD
Mostoles (Madrid)

Vendo Spectrum 48K, buen estado. Cables, alimentación y manuales. Regalo muchos programas (juegos y utilidades) y revistas. 20.000 ptas. negociables.

Pablo Pérez de Ayala González
Martínez Campos 24, 3ºD
Tel. 25 72 75
18002 Granada

Vendo ZX-Spectrum Plus (quince dias reales de uso), más interface, joystick, libros, revistas, juegos (Comando, Rambo y 28 juegos más). Completamente nuevo, por 32.000 ptas. Llamar a:

Juan Manuel Joga
Tel. (91) 463 00 51

Vendo consola video juegos Philips G-7000. Con tres juegos, por el precio a convenir. Escribir a:

Sergio Sackman
Manigua, 14, 1º 1ª
Tel. (93) 351 38 83
08027 Barcelona

Vendo video juegos PHILIPS G 7400 en muy buenas condiciones con dos mandos y juegos incluídos por 15.000 pts.Para cualquier información llamar al:

Tel.216 64 65

Intercambio programas.Preferiblemente utilidades;aunque también juegos de Spectrum. Novedades. Preguntar por:

Jose Manuel Nicolás
Calletano Vinzia, 80-5º-3ª
Tel. (93)593 06 34.

Cambio programas de juegos con gente de toda España.Poseo últimas novedades.Interesado llamar a:

Javier
Tel. (985)39 12 18
Jorge
Tel.(985)39 27 14

Vendo Lápiz optico con interface y su correspondiente programa. En perfecto estado, por 4.000 pts.

Jaime Solá
San Magín, 42
Tel. (93) 893 27 41
Vilanova I la Geltru (Barcelona)

Intercambio mapas, instrucciones, etc. Interesado en el código de Dragontorc. También intercambio programas de utilidades.

Antonio Fernanez
Cristobal Colom, 57-61
Santa Coloma de Gramanet
(Barcelona)

Se vende QL impresora y monitor monócromo por 120.000 ptas., negociables. Todo está en muy buen estado.

Juan Carlos
Tel. 413 90 13 de 14.30 a 15.30
Madrid

Intercambio programas para el ZX Spectrum 16K o 48K.

Carlos Lozano Barcelo
Braulio Lausin, 10, 5ºF
50008 Zaragoza

Vendo QL con los programas Forth, Pascal, Toolkit, Ensamblador, Desensamblador, Sprite, Ajedrez y Tenis. Todo por 65.000 ptas. También interface 1 y microdrive para Spectrum por 20.000 ptas.

Joaquín Fuster Sierra
Colón, 7
46004 Valencia

Intercambio juegos para Spectrum 48K. Títulos comerciales como: Rambo, Sir Fred, Comando, etc., desearía contactar con usuarios de toda España a ser posible de la región valenciana. Escribir o llamar:

Cais, 11
Santiago Frasquet
Tel. (96) 280 53 56
Villalonga (Valencia)

Cambio programas para el Spectrum. Desearía conseguir: Lords of Midnight, Saimazoon, Babaliba, Dum Darach. Tengo buenos juegos.

Antonio Clemente Meco
Tel. 455 16 71
Madrid

Vendo ZX Spectrum Plus completo, con teclado en español (sin estrenar, comprado estas Navidades). Regalo compilador Pascal y 25 programas de juegos. (32.000 ptas.).

José
Tel. 637 24 02 (Madrid)
Tel. 24 38 27 (Salamanca)

Estoy interesado en conseguir los programas: Doomdark's Revenge y D.Day. Escribir a:

Ignacio López de Torre
Calle del Río, 27
Miranda de Ebro (Burgos)

Vendo o cambio por un televisor b/n pequeño, videojuegos para TV en color, con cuatro juegos y sus dos joysticks. 6.000 ptas. Escribir a:

Luis Alberto Cano
Doctor Manzanares, 20, 1ºD
Ocaña (Toledo)

DESCUBRE AL ASESINO

Muchos abandonaron en el camino, pero otros (pocos) llegasteis hasta el final en el reto planteado, Finalmente hubo que realizar un sorteo, imparcial por supuesto, al que asistimos elementos de la redacción de INPUT y personal de Anaya Multimedia.
Los mensajes planteados tenían como respuesta lo siguiente:

"ES UNA VIOLENCIA FINGIDA PUES SE TRATA DE UN TIPO DE CAJA INACCESIBLE A CUALQUIER TIPO DE PALANCA".

" LAS ATADURAS NO SON DE UN PROFESIONAL DA LA IMPRESION DE QUE TENIAN MIEDO DE LESIONARLE".

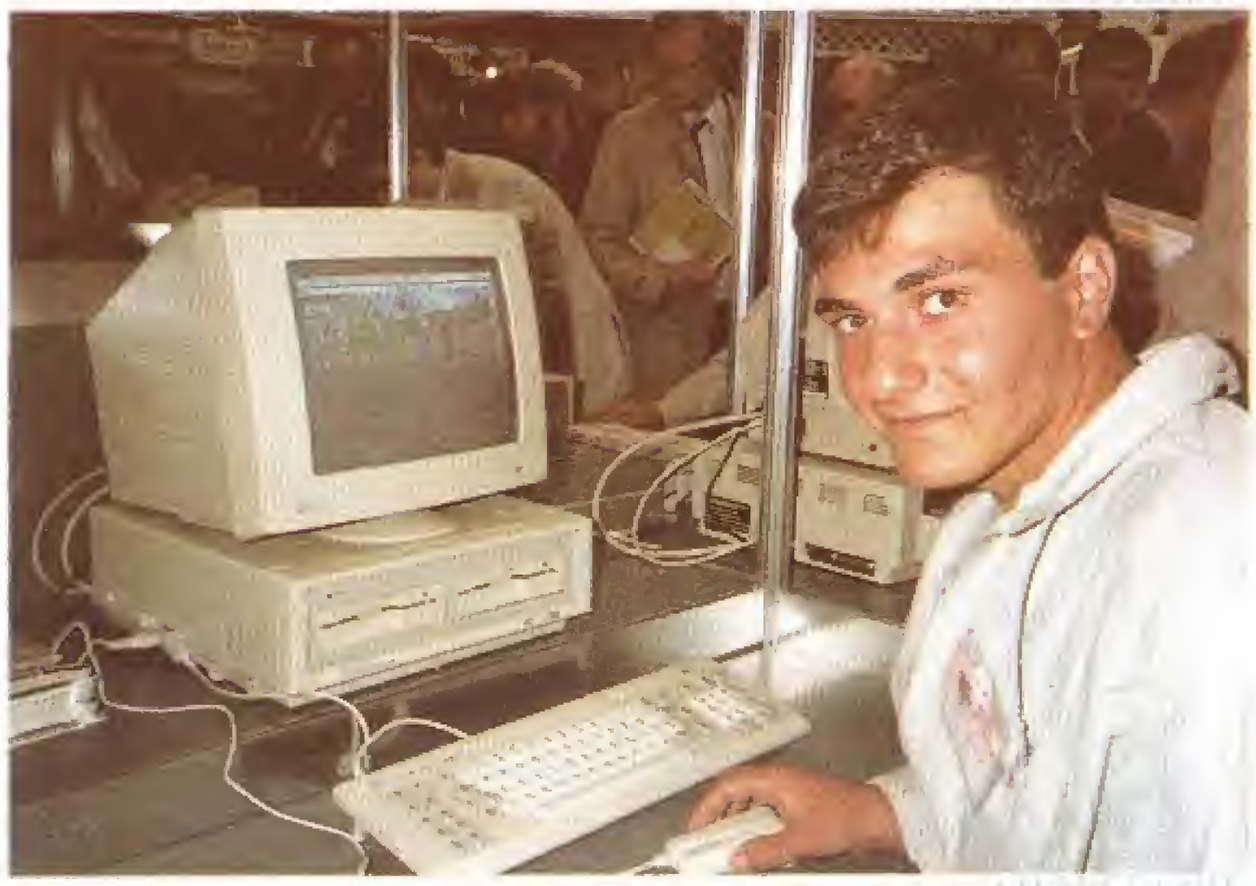
"TIENE LA COSTUMBRE DE MORDER LAS COLILLAS DE LOS CIGARROS QUE FUMA SIN PARAR HASTA DESHACERLAS".

"EN EL REGISTRO EN LA CASA DE LA Victima aparecieron las joyas habilmente CAMUFLADAS BAJO EL SUELO DEL DORMITORIO".

"A PESAR DE ESTAR BASTANTE QUEMADA EN LA COLILLA SE PODIAN DISTINGUIR PERFECTAMENTE LAS MARCAS DEJADAS POR LOS INCISIVOS DE UN FUMADOR NERVIOSO".

El intrépido y sagaz descifrador de claves se llama Manuel Bautista López. Todo un hallazgo para los servicios de inteligencia de cualquier pais, que no dudamos se lo rifarán cuando este ejemplar caiga en sus manos. Todo un criptofuturo por delante.

En la fotografía podemos apreciar a nuestro afortunado Manuel disfutando de su estancia en el PCW Show de Londres. Le acompana Santos Rodriguez, director de producto y delegado de Anaya Multimedia para el significativo evento.



EL ZOCO

Importante. Ha surgido un club de usuarios de Spectrum para intercambio de juegos. Se poseen buenos como: Camelot, Exploding Fist, I of the Mask, entre otros. Interesados llamar o escribir:

José Juan Zapater
Avda. Aragón, 61, 5°A
Tel. (974) 83 03 90
Alcañiz (Teruel)

Multiface 1. Vendo este importante periférico que le permitirá copiar cualquier programa de Spectrum. Además va provisto de RESET y SALIDA DE VIDEO. Precio 10.000 ptas. y le acompaña certificado de garantía por 6 meses. Conectar con:

José R. Andicoechea Rodrigo
Ciudad de Aracena, 7
21001 Huelva

Vendo ZX Spectrum Plus 64K, adjunto 20 programas comerciales últimas novedades; inmejorable estado. Se admite correspondencia para intercambio de programas.

Manuel Vitureira Suárez
Travesía de Vigo, 54, 4° dcha.
Vigo 6 (Pontevedra)

Regalo más de 70 revistas para Spectrum por la compra de un Spectrum 48K, un cassette especial marca «Computone» y una amplia cantidad de juegos (los últimos para Spectrum). Todo por 45.000 ptas. Interesados escribir o llamar a:

Arketale, 11, 2°
Tel. (943) 76 44 71
Bergara (Guipúzcoa)

Deseo conectar con usuarios del ZX Spectrum Plus, para intercambiar ideas y sobre todo para que me puedan resolver dudas. Escribir a:

Mª del Mar Rodriguez Blázquez
Camarena, 312, B3
28047 Madrid

Vendo Interface joystick programable Indescomp, garantía en blanco por 3.900 ptas.

Pedro Saez
Jacinto Benavente, 2
Pilar de la Horadada (Alicante)

Vendo ordenador ZX Spectrum 48K en 20.000 ptas. Con garantía Investrónica. Regalo programas. Vendo Interface 1 + Microdrive en 25.000. Regalo cartuchos. Llamar:

Bayon
Tel. (985) 29 21 87

Vendo Spectrum Plus completamente nuevo con los 50 mejores juegos del mercado. Precio 32.000 ptas. Además vendo los tres mejores copiones, Trans-Express, Kapi Copy 5, Kapy Copy 6.

Alfonso
Tel. (93) 204 30 22

LISTADO DE ENSAMBLADOR PARA MANEJO DE INTERRUPCIONES

En el número anterior de **INPUT** veíamos cómo utilizar las interrupciones en el Z-80, y proponíamos un divertido programa musical, del cual publicamos la versión en BASIC. Sin embargo, el listado en código máquina, cuyas principales rutinas se analizaban en el texto, se nos quedó fuera. Este era:

```
 10         ORG  64505
 20 ORIG-D  DEFW 64640
 30 POSIC   DEFW 0
 40 CAMBIO  DEFB 0
 50 DURAC   DEFB 0
 60 VECTOR  DEFW 0
 70 INTERR  PUSH IX
 80         PUSH HL
 90         PUSH BC
100         PUSH DE
110         PUSH AF
120         JP   ARRANQ
130 FIN     POP  AF
140         POP  DE
150         POP  BC
160         POP  HL
170         POP  IX
180         JP   56
190 ON      LD   HL,INTERR
200         LD   (VECTOR),HL
210         DI
220         LD   A,251
230         LD   I,A
240         IM   2
250         LD   HL,(ORIG-D)
260         LD   (POSIC),HL
270         LD   A,1
280         LD   (CAMBIO),A
290         DEC  A
300         LD   (DURAC),A
310         EI
320         RET
330 OFF     DI
340         IM   1
350         EI
360         RET
370 ARRANQ  LD   A,(CAMBIO)
380         CP   0
390         JP   Z,FIN
400         LD   A,(DURAC)
410         CP   0
420         JP   Z,NOTAS
430         DEC  A
440         LD   (DURAC),A
450         JP   FIN
460 NOTAS   LD   HL,(POSIC)
470         LD   A,(HL)
480         CP   255
490         JP   Z,REST
500         LD   (DURAC),A
510         INC  HL
520         LD   A,(HL)
530         INC  HL
540         LD   E,A
550         LD   A,(HL)
560         INC  HL
570         LD   (POSIC),HL
580         LD   H,A
590         LD   L,E
600         LD   DE,4
610         CALL 949
620         DI
630         JP   FIN
640 REST    LD   HL,(ORIG-D)
650         LD   (POSIC),HL
660         LD   A,1
670         LD   (CAMBIO),A
680         DEC  A
690         LD   (DURAC),A
700         JP   FIN
```

CURSO DE BASIC + MICROORDENADORES

prácticas con...

Microordenador ZX SPECTRUM

Microordenador COMMODORE

Microordenadores AMSTRAD, MSX, PC

Para saber cómo hablar con los ordenadores

El Curso CEAC a Distancia, BASIC + Microordenadores, le va a introducir paso a paso, con un cuidado método, en uno de los temas más apasionantes de nuestros días: **la programación de ordenadores.**

Al aprender PRACTICANDO desde un principio a programar BASIC, lenguaje diseñado especialmente para dar los primeros pasos en programación, estará sentando las bases para el estudio de cualquier otro lenguaje de alto nivel.

Curso CEAC de BASIC + Microordenadores: un diálogo permanente con el ordenador.

CEAC

CENTRO DE ENSEÑANZA A DISTANCIA
AUTORIZADO POR EL MINISTERIO DE EDUCACION Y CIENCIA N.º 8039185
(BOLETIN OFICIAL DEL ESTADO 3-6-83)
Aragón, 472 (Dpto. **T-ZE**) 08013 Barcelona
Tel.: (93) 245 33 06

Otros Cursos:

- Introducción a la Informática
- Electrónica (con experimentos)
- Contabilidad
- Fotografía
- Curso de Video
- Decoración

ESTAS ENSEÑANZAS SE AJUSTAN AL ART. 35
DEL DECRETO 707/1976 Y A LA ORDEN MINISTERIAL DE 5/2/1979

GRATUITAMENTE

Actúe ahora en su propio beneficio y pídanos información.

Sí, deseo recibir a la mayor brevedad posible información sobre el Curso de: _____

_____

Nombre y apellidos _____ Edad _____

Domicilio _____

_____ N.º _____ Piso _____ Pta. _____ Tel. _____

C. Postal _____ Población _____

_____ Provincia _____

Profesión _____

CEAC. Aragón, 472
(Dpto. T-ZE) 08013 Barcelona

o llame... (93) 245 33 06 de Barcelona

GHOSTS'N GOBLINS

Officially Licenced Coin-Op Classic from CAPCOM

¡SUPERPROMOCION!

¡EL MONSTRUO MILLONARIO!

COMPRA TU PROGRAMA GHOSTS'N GOBLINS
Y GANA MUCHOS MILES DE PTAS.

BALL CHIP

SPECTRUM
AMSTRAD
AMSTRAD DISK
COMMODORE

EL SUPER EXITO

Ghosts 'n Goblins es la auténtica versión para ordenadores domésticos del clásico juego de arcade de las máquinas de moneda de Capcom, autores de Super-éxitos mundiales como Commando.
Ghosts 'n Goblins es la clásica historia fantástica donde el Caballero debe rescatar a su Dama de las garras de las criaturas del mal.
Con unos maravillosos efectos y gráficos, técnicamente excelentes, este juego es claramente un Núm. 1.

elite

ZAFIRO SOFTWARE DIVISION
Paseo de la Castellana, 141. 28046 Madrid.
Tel. 459 30 04. Tel. Barna. 209 33 65.
Telex: 22690 ZAFIR E